Do agents dream of abiding by the rules?

Learning norms via behavioral exploration and sparse human supervision

Peter Fratrič
Informatics institute,
University of Amsterdam
Amsterdam, The Netherlands
p.fratric@uva.nl

Mostafa Mohajeri Parizi*
Informatics institute,
University of Amsterdam
Amsterdam, The Netherlands
m.mohajeriparizi@uva.nl

Giovanni Sileno*
Informatics institute,
University of Amsterdam
Amsterdam, The Netherlands
g.sileno@uva.nl

Tom van Engers
Leibniz Institute,
TNO/University of Amsterdam
Amsterdam, The Netherlands
t.m.vanengers@uva.nl

Sander Klous
Informatics institute,
University of Amsterdam
Amsterdam, Netherlands
klous.sander@kpmg.nl

ABSTRACT

In recent years, several normative systems have been presented in the literature. Relying on formal methods, these systems support the encoding of legal rules into machine-readable formats, enabling, e.g. to check whether a certain workflow satisfies or agents abide by these rules. However, not all rules can be easily expressed (see for instance the unclear boundaries between tax planning and tax avoidance). The paper introduces a framework for norm identification and norm induction that automates the formalization of norms about non-compliant behavior by exploring the behavioral space via simulation, and integrating inputs from humans via active learning. The proposed problem formulation sets also a bridge between AI & law and more general branches of AI concerned by the adaptation of artificial agents to human directives.

CCS CONCEPTS

• **Computing methodologies** → **Modeling and simulation**; **Artificial intelligence**; **Learning settings**; • **Applied computing** → *Law*.

KEYWORDS

Norm identification, Norm induction, Normative systems, Compliance checking, Non-compliance.

ACM Reference Format:
Peter Fratrič, Mostafa Mohajeri Parizi, Giovanni Sileno, Tom van Engers, and Sander Klous. 2023. Do agents dream of abiding by the rules?: Learning norms via behavioral exploration and sparse human supervision. In *Nineteenth International Conference on Artificial Intelligence and Law (ICAIL 2023), June 19–23, 2023, Braga, Portugal*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3594536.3595153

*Both authors contributed equally to this research.

1 INTRODUCTION

Compliance is crucial for sustainable business, e.g., in banking, healthcare, (international) trade, construction, logistics, and in any process in which (potentially sensitive) data is shared. Failure to comply with laws and regulations can not only have severe consequences on individual organizations, but also have destructive influence on the economy. Organizations employ teams of experts to prevent consequences of setting up improper business processes, as well as to monitor and respond to non-compliance by other social participants with whom organizations engage (e.g. for contractual agreements, possibility of frauds, legal accountability, due diligence). In order to detect non-compliance, researchers and analysts have introduced and applied over the years various computational methods, ranging in the whole AI spectrum, from symbolic (rule-based systems, knowledge graphs) to sub-symbolic approaches (machine learning models). Automation of these processes have proved to increase the efficiency of compliance-related processes, but a number of challenges remains to be addressed (see e.g. [2, 14]).

The present work focuses in particular on the problem of *identifying norms* that are not made fully explicit by the legal system (and thus cannot be represented directly using a formal language). Whether we speak of tax compliance, environmental carbon rules or trading regulations, the computational encoding of legal rules that are relatively vaguely defined, and by nature rely on the ruling of legal authorities on individual cases, remains an open question. Yet, assuming consistency in the rulings and in the behavioral patterns associated with non-compliance, one can expect the emergence of a norm describing the set of non-compliant behaviors. The main research question of this study is how to get from (a minimal number of labelled) cases to a computational encoding of behavioral rules defining non-compliance. Building on the intuition that a norm violation is usually visible as a complex, but still highly regular phenomenon, our contribution focuses on norm identification and induction by means of *simulation* and *utility maximization*, setting up an *active learning* loop in which an oracle (e.g. a legal expert) is asked to qualify behavior deemed relevant of attention. This problem formulation can also be easily associated to other branches of research in AI, concerned by the adaptation of behavior of artificial agents (e.g. robots) to directives given incrementally by humans.

Related works. In recent years, several computational normative systems have been presented in the literature. Based upon formal methods, these systems support the encoding of legal rules into some machine-readable format (e.g. [11, 14]), enabling e.g. to check whether a certain workflow satisfies or agents abide by these rules (or at least their computational version). Besides formal validity, these systems are mostly concerned by providing a relatively user-friendly language/framework for normative specification [18, 26, 31, 36], computational scalability, but in most cases do not concern themselves with using behavioral data. Problems arise however in presence of rules which are difficult to formalize, for instance because they are inherently vague. The associated *implicit legal rules* span an infinite set of various behavioral patterns and legal structures, that can be found for instance in the area of tax optimization or stock exchange regulation. A system used to find an optimal sequence of actions satisfying the formalized *explicit legal rules* as constraints, but likely violating the implicit rules, can be used as a very efficient tool for "breaking" the law. Computational methods can be used to limit this possibility, by facilitating the alignment of computational normative systems to what is yet implicit in the legal system.

The area of research in which this study can be positioned is *norm identification* and *norm induction*, which has been predominantly studied in the area of autonomously learning agents [7, 28, 29]. Symbolically represented behavior learned from observations have been investigated using various approaches combining symbolic and sub-symbolic methods, e.g. Bayesian approach [9], grammatical version of Markov chain Monte Carlo sampling [8], or probabilistic inductive logic programming [35]. The last two approaches were used in the area of robotics, where frameworks exploring human guidance to agent's normative compliance are actively developed [10, 22]. Our study takes an analogous approach: humans provide feedback information to the norm extraction system on whether an agent is complying with implicit legal rules.

The general task can be characterized as aiming to go from cases to rules while maintaining consistency. The inductive problem of identifying the decision patterns applied by a decision-maker on individual cases partially overlaps with the traditional *case-based reasoning* research track in AI & Law [3, 27], aiming to reconstruct the structure of the *rationale* behind case decisions, typically identifying relevant factors and their relative contributions to the conclusion; however, the "behavioral definition" issue studied in our work focuses primarily on capturing legally relevant behavioral patterns (*scripts*), rather than relevant contingent factors: the temporal nature of actions will play a major role.

Contribution. The paper introduces a framework for the automated formalization of norms by exploring the behavioral space via simulation, under assumptions (e.g. existence of utility function) that are likely to hold in certain behavioral domains (e.g. complex financial transactions). Inputs from humans are integrated in an active learning cycle, such that the framework provides feedback suggestions by calculating uncertainty using modern transduction learning methods [24]. This research outcome can serve as an entry point to encourage deeper connections between the area of AI & law and contemporary research in autonomous artificial agents.

Structure of the paper. This study is structured as follows. Section 2 provides a few relevant examples of non-compliant behavior. In section 3 we consider a normative system as a transition system, and define a utility function, and discuss the need for active and semi-supervised learning component. In section 4, we propose an active learning framework using (ideally) a small number of labeled instances of norm (non-)violation. The application of the framework is illustrated in section 5 on a practical example, as well as on a simulation example. The simulation example implements a path finding agent in a normative grid environment using reinforcement learning. We choose this environment not only because it is a rather classical problem in the area of autonomous agents, but also because a number of concepts introduced in this study admit with it a natural and easy to visualize interpretation. Towards the end, we discuss the practical applicability, the limitations of the framework, and a brief note on future work.

2 MOTIVATING EXAMPLES

In socio-legal terms, a *wrong* supposedly occurs when there are unmet social expectations (for at least one of the participants). These expectations are generally specified (in legislation, organizational policies, agreement, contracts, etc.) via normative directives, typically concerning prohibitions, obligations, and permissions. In principle, a wrong should be detectable from the violation of normative directives, but practical domains exhibit different types of failure [33]. For instance, in the context of taxes due for real state transactions, the involved parties may: (a) not pay taxes (*syntaxic, qualitative failure*), (b) pay a formally wrong amount (*syntaxic, quantitative failure*), (c) pay a formally correct but not "right" amounts, for instance by setting a much-lower nominal price following a fraudulent swap-scheme (*semantic failure*). Indeed, the unclear boundaries between tax planning, tax avoidance, and also the role of tax havens trigger debates both in the academic literature and in policy circles [23], offer prototypical domains of application for our study.

The scheme named Double Irish–Dutch Sandwich is for instance a notoriously known base-erosion and profit-shifting corporate tax tool, that was legal until 2015. It takes advantage of a legal loophole that allows royalty payments to be made to several offshore tax havens (like Bermuda), without incurring Dutch withholding tax. In the global banking system, this activity can be visible not only by a regular streams of transactions, but also by a complex ownership network structures [13]. Although the principles of how this specific scheme operates are known, one can imagine that there are many such schemes with operational principles known only to their users.

Theoretically, the high behavioral regularity associated with a given scheme (to reach the desired profit) could be used to identify the norm that may close the relevant loopholes. This can be done by having a human analyst identify the loopholes and design a new law that will label the scheme illegal. Alternatively, one can take examples of behavioral manifestations of the scheme, use some automated procedure to identify the general pattern of actions, and define a new norm by banning every behavior matching the pattern.

In general, in any practical normative system there might be instances of behavior that are undesired, or illegal according to a rule that was not (yet) included into the system; we call the associated norm, because of its unknown formal representation, *implicit rule*. Conversely, all norms with known (or adequate) formalization, integrated and enforced in the normative system, are called *explicit rules*.

Under these general terms, the problem may be abstracted from specific legal domains, and connected to other branches of AI. For instance, in discrete grid environments, *allowed/prohibited* schemes would correspond to allowed/prohibited regions on the grid. Known constraints (explicit rules) would be reified as walls or other blocking items, and the problem targeted here be to identify not-yet reified social constraints (implicit rule).

3 PROBLEM FORMULATION

This section introduces formal requirements necessary to define the problem in a mathematical way. We elaborate on an abstract problem formulation, the requirement and use of a utility function, and of active and semi-supervised learning.

3.1 Modelling the normative system

Consider a computational system that implements the formalization of explicit rules in some legal domain. Abstracting away from the specific programming language, we consider it to be a transition system $TS = (S, A, \rightarrow, s_{start}, s_{end})$ where

- S is a set of *states*
- A is the set of actions
- $\longrightarrow \in S \times A \times S$ is a transition relation
- $s_{start} \in S$ is the initial `start` state
- $s_{end} \subset S$ is the set of terminating `end` states

Executions of this TS describe possible behavioral processes occurring in the domain. For the normative (prescriptive) dimension, we can distinguish **hard constraints** and **soft constraints**. Let us denote $\mathcal{P}(TS)$ to be a set of all processes generated by the TS. The hard constraints $\mathcal{H} \subset \mathcal{P}(TS)$ implement explicit legal rules into the system by removing specific processes from the set of possible processes. Sequences of actions that are possible to execute in the TS, but violate the implicit rule, are members of the set of non-compliant behaviors Γ, and are regarded as soft constraints. One can easily see that $\mathcal{P}(TS) = \Gamma^c \cup \Gamma$: every process generated by the transition system either complies to the implicit rule or not. However, unless one knows all the members of Γ or Γ^c, or possess a predicate defining which instances of $\mathcal{P}(TS)$ belong to one or the other set, it is not possible to enforce compliance to soft constraints in an exact manner. From a legal point of view, soft constraints can be enforced only *ex-post* for each case individually, unlike the hard constraints that are enforceable *ex-ante* or on runtime.

Let us assume that, for every transition, the agent incurs a cost of transition, well-defined for each state, $R : S \times A \times S \longrightarrow$ R. In addition, we define a **utility function** $u : \mathcal{P}(TS) \longrightarrow$ R, that aggregates individual costs for the whole sequence from the start `state` to the `end` state. This utility function must be however viewed not as exclusively concerned with the transition system, but also encompassing a wider horizon of effects, including the risk of detection, or expected actions of other agents that may influence the utility of a sequence. u captures therefore an aggregated value; expected value would then be a formulation consistent with this approach, as well as other models or heuristics used for the actualization of value. If more actors are influencing the states of the system, then the actions of these actors can be considered as a stochastic component of the system, and extend the definition of the utility as an expected value over these stochastic variations (one could then extend the model from a transition system to a Markov decision process). Here we will restrict our attention to a deterministic single-agent setting, but we will return to the discussion about the multi-agent case later in the paper.

3.2 Exploration of the behavioral space

Taking the point of view of a potential perpetrator, one can consider an optimization problem of finding a sequence of state transitions with the highest utility, disregarding the soft constraints. This can be done both using state-of-art planning algorithms, or model-free approaches such as reinforcement learning or ant colony optimization. More formally, we may write:

$$\begin{aligned} &\max_{x \in \mathcal{H}} u(x) \\ \text{s.t.} \quad &start(x) = s_{start} \;\text{ and }\; end(x) \in s_{end} \end{aligned} \tag{1}$$

where $\mathcal{H} \subset \mathcal{P}(TS)$ denotes sequences that satisfy the hard constraints, and $start(x)$ and $end(x)$ are operators returning the initial state and end state, respectively. This optimization process only respects the hard constraints that are enforceable in the system; if needed, the agent may violate the soft constraints to find sequences of higher utility.

3.3 Norm identification

Formalizing the soft constraints to be integrated within the system as hard constraints would stop the agent to find profitable sequences that violate the soft constraints. With this motivation, let us define the problem of norm identification for the transition system. We select a grammar G that defines a search space over expressions that can describe the set of cases in Γ, and then consider an optimization routine to maximize the number of covered elements in Γ and at the same time minimize the number of covered elements in Γ^c.

Let us denote $g \in G$ to be an expression taken from the grammar G, and let $\mu_g(X)$ be a set function that counts the number of elements of $X \subset \mathcal{P}(TS)$ that are covered by the expression g. Then we may formulate the optimization problem of finding an expression $g \in G$ that optimally covers the set Γ as:

$$\max_{g \in G} \mu_g(\Gamma) - \mu_g(\Gamma^c) \tag{2}$$

A constraint on $\mu_g(\Gamma^c)$ can be introduced to prevent covering too many instances of Γ^c. This requirement can be supplemented by a complexity penalty to set a preference on parsimonious representation. Equivalently, one could aim to find the description of the set Γ^c, and try to minimize the cover over the elements of Γ, but for reasons discussed in the section 4, this should be less effective.

3.4 Active and semi-supervised learning

The exact value of the measures $\mu_g(\Gamma)$ and $\mu_g(\Gamma^c)$ for expression $g \in G$ cannot be realistically known, because one does not have a list of all elements of Γ or Γ^c. The value of μ_g will always be an estimate, depending on the available information. Let us consider $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_{-1} \cup \mathcal{D}_0$ to be a set of instances of $\mathcal{H}$, respectively labeled as compliant $\mathcal{D}_1$, non-compliant $\mathcal{D}_{-1}$, and unlabeled $\mathcal{D}_0$. In full generality, one needs to address two questions. Firstly, which elements of $\mathcal{H}$ to sample. Secondly, which sampled elements to label. Both of these questions are extensively studied in the area of *active learning*, where budgetary constraints are introduced for both the sampling and labeling process [30].

The answer to the first question is provided here by means of the utility-maximizing agent solving the optimization problem (1). The method we propose finds the local optimum of the utility function with its locally optimal sequence, and use a *perturbation method* [16] to obtain instances of behavior that are structurally close to the optimal sequence, i.e. obtaining representative examples of a fraud scheme (see section 4.2 for further details).

The answer to the second question is more complex. In the area of active learning, the oracle usually represents the human component capable of labeling the sampled instances. From the legal point of view, the oracle can be e.g. a team of legal analysts, or a judge, that decides if an instance complies to the implicit legal rule, thus the soft constraint, or not. The label presented to the oracle are selected depending on a query strategy, prioritizing which instances to label such that (in principle) the highest information gain is obtained with respect to the optimization routine (2). The development of general strategies is an active area of research, however, here we can design the strategy based on theoretical considerations.

In the application context, the budgetary constraints on the number of queries might be extremely tight. This means $\mathcal{D}_0$ will have much larger cardinality than $\mathcal{D}_1 \cup \mathcal{D}_{-1}$. This not only means that the estimate of $\mu_g(\Gamma)$ and $\mu_g(\Gamma^c)$ might have a high estimation error, but also that the number of instances might be too low to find an expression g that realistically covers a significant part of Γ. In general, the area of research concerned with using unlabeled data is *semi-supervised learning* [37]. Transduction learning [38] is a method of semi-supervised learning related to a set of algorithms known as *case-based learning* [1]. This method avoids solving the general problem of finding the mapping between the input and the labels, but rather aims to obtain the labels by investigating the instances locally. Methods operating locally are preferred in this study, because global classifiers typically require a high number of labels. We elaborate on the locality property in section 4.

4 COMPLETION OF THE NORMATIVE SYSTEM

This section presents a framework that is combining the optimization problems (1) and (2) with an active and semi-supervised learning component in order to formalize soft constraints and integrate them into the system as hard constraints. The framework is visualized on the diagram on Figure 1.

4.1 Initialization

As the initial step, we need to define the transition system: the environment in which the agent will operate, the hard constraints, the utility function, the `start` state, and the `end` states. The whole process is valid per specific `start` and `end` states. For example, if the agent owns \$1000 at the start, then the extracted norm will be valid only for the given `start` state.[1]

[1]Alternatively, with additional machinery, one may consider a variable defining the amount, but this is out of the scopes of the present paper.

4.2 Iterative process

The proposed framework requires certain structural properties of the set Γ to be likely true, justifying the choice of using the utilitarian agent, as well as the perturbation method. These properties allow for the diagram on Figure 1 to form a closed loop, with explicit termination criteria. For the cases ruled as norm-violating, we often see in practice that illegal activities have a tendency to cluster into distinct schemes that follow a certain behavioral pattern (see e.g. [33]). The combination of consistency in legal practice and behavioral pattern clustering support us in assuming a partitioning of the Γ into subclasses of non-compliance Γ_k such that $\Gamma = \Gamma_1 \cup ... \cup \Gamma_n$, with relatively small intersection between individual classes. If we wish to find a formal representation for the set Γ, it seems reasonable to find the representation for each class Γ_k separately. The main reason for this is that each class should correspond to a local optimum of the utility function. This intuition was not only observed empirically [4, 5, 19, 25, 32], but was also demonstrated computationally, e.g. for Ponzi schemes and for tax evasion [12, 15].

4.2.1 Structural neighborhood. Following up on the previous discussion, we are interested to let the agent converge to different local optima, select the set of instances for automated formalization, and repeat until all classes are covered, ideally starting from the most profitable ones. In other words, the agent needs to traverse the classes Γ_k. Without loss of generality, one can assume that each class Γ_k has an element with the highest utility. Let us denote γ_k to be the local optimum of the class Γ_k with utility $u(\gamma_k)$, i.e. the most profitable noncompliance scheme of the class. If we consider a neighborhood function $\mathcal{N}$ that assigns a set of neighbors $\mathcal{N}(\gamma_k) \subset \mathcal{P}(TS)$, then we can pose a question about the intersection $N(\gamma_k) \cap \Gamma$, that is, what kind of perturbation of γ_k can change the normative membership of the behavioral instance. Clearly, one does not know how to decide which instances of $\mathcal{N}(\gamma_k)$ are members of Γ_k, because this is equivalent to finding the formal expression for the class Γ_k, which is the primary objective. Nevertheless, if the local optimum γ_k can be found, then one can expect a significant intersection between $\mathcal{N}(\gamma_k)$ and Γ_k, which gives us an idea of how to generate elements of Γ_k.

4.2.2 Perturbation of the local optimum. In order to obtain a formalization of the class Γ_k, one needs to build the training set $\mathcal{D}$ that includes not only the optimal instance, but also instances in its proximity. If the agent ends up converging to a local optima by solving the optimization problem (1), then we obtain the locally optimal transition sequence γ_k. A way to obtain the neighborhood $\mathcal{N}(\gamma_k)$ is to perform perturbations to γ_k. In general, if we obtain a random perturbation of γ_k, then one needs to check the compliance of this new instance with the oracle, a choice that would bring to violate the budgetary constraint, as the number of queries would be too high. For this reason, a method needs to be defined that prioritizes queries to produce the highest information gain for solving the optimization problem (2).

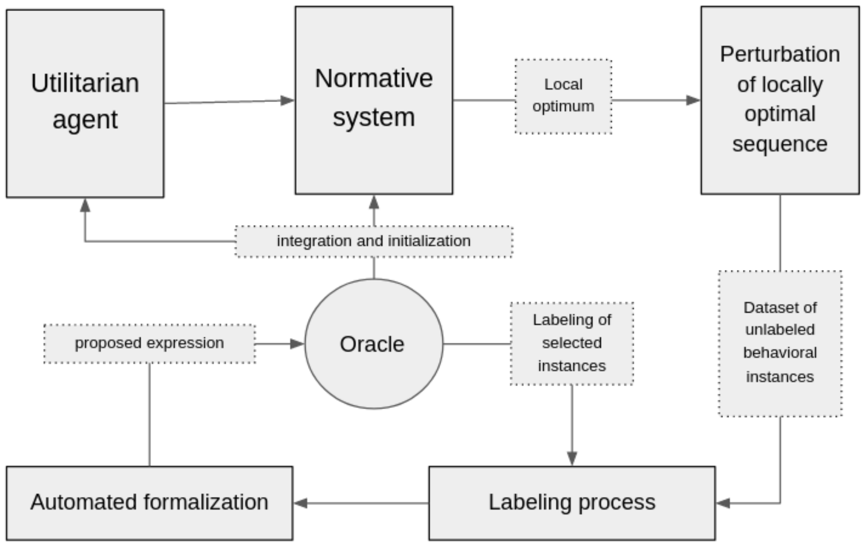

Utilitarian agent
Normative system
Local optimum
Perturbation of locally optimal sequence
integration and initialization
proposed expression
Oracle
Labeling of selected instances
Dataset of unlabeled behavioral instances
Automated formalization
Labeling process

Figure 1: The process starts by introducing the utilitarian agent to a normative system, where both hard and soft constraints are present, to solve the optimization problem (1). After the agent converges to a local optimum, the locally optimal trajectory will be used to initiate a perturbation process in order to generate instances of behavior that are structurally close to the trajectory. A labeling process is then deployed to obtain the dataset $\mathcal{D}$. The dataset is used to find a formal expression by solving the optimization problem (2), defining non-compliant instances in the neighborhood of the locally optimal trajectory. The human analyst, acting as an oracle, analyzes the proposed expression, and either terminates the process or initiates a new iteration.

4.3 Labeling process and automated formalization

Depending on how samples are obtained, active learning is usually divided into a *stream-based selective sampling* and *pool-based sampling* [30]. We assume that the process of obtaining the unlabeled instances of $\mathcal{N}(\gamma_k)$ is separated from the oracle, which means we position ourselves in the latter approach.

4.3.1 Level of active involvement. Since the optimization problem (2) depends on the quality of data $\mathcal{D}$ provided, results may vary. Active learning often takes advantage of this by evaluating the query strategy based on the optimization results, possibly changing the strategy if the performance is not sufficient. This means that the number of repetitions of the optimization problem (2) can be relatively high, and depends on computational resources. Searching the discrete space of the grammar G is however a computationally intense task even for relatively simple grammars. For this reason, we assume that for each class Γ_k the optimization process (2) is solved only once, which means the emphasis is placed on gaining information from unlabeled instances.

4.3.2 Selection based on similarity. The challenging aspect of norm induction is that arbitrary large perturbations from γ_k do not necessarily change the compliance class membership, and, at the same time, even one change in a single state-action pair can make γ_k fully compliant (see e.g. section 5.1). Due to this property, we claim that the query method ought to be focused on the boundary region, that is, the instances where the compliance class is shifted by the perturbation. Therefore, we need a method that provides a good guess where the boundary region is, i.e. where the instances with the highest level of uncertainty about the class membership are located. Case-based learners typically utilize a similarity function to obtain numeric values describing the similarity between instances. The working assumption is that instances with high similarity are more likely to have the same label.[2] A *weight matrix* W can be obtained by calculating the similarity between each pair of the sample set $\mathcal{N}(\gamma_k)$. A transductive inference method can be then applied to calculate scores $\mathbf{w}$ of unlabeled instances. In binary classification, a score close to 1 would indicate compliance, and a score close to -1 non-compliance. Labeled instances have scores equal exactly to 1 or -1. The score close to zero might be the most interesting one for the query strategy, as zero indicates either an instance that is not similar to most data, or an instance that lies on the boundary.

4.3.3 Forming of the final dataset. The labeled sets $\mathcal{D}_1 \cup \mathcal{D}_{-1}$ provided by the oracle might be insufficient to have a good estimate of $\mu_g(\Gamma) - \mu_g(\Gamma^c)$. Moreover, the correct expression g from the grammar G might not be identifiable for a small labeled dataset. To mitigate this issue, the score vector $\mathbf{w}$ can be element-wise compared to a predefined threshold η and enrich the set of labeled instances. In other words, the i-th generated instance by the perturbation method will be added to the set $\mathcal{D}_1$ if $w_i > \eta$ or to the set $\mathcal{D}_{-1}$ if $w_i < -\eta$. Finally, the enriched dataset is then used to solve the optimization problem (2) for the chosen grammar G.

[2]For the moment, we simply assume that the similarity function is given. We will discuss this assumption in the examples and in the discussion section.

4.4 Termination, verification and testing

The human analyst can terminate the process on Figure 1 based on predefined termination criteria. The trivial criterion for termination is that the dataset $\mathcal{D}_{-1}$ is empty (no soft constraint violation was observed). Once the system generates a proposed formal expression, there are reasons why the oracle may reject the expression. It can be that the proposed expression blocked every possible path from s_{start} to s_{end}, labeling the system unusable. If the expression is possible to understand by the human analyst, then the analyst acting as the oracle can inspect the proposed expression and make the final decision by conducting its legal analysis.[3] If the proposed expression is accepted, then it will be integrated into the normative system as a hard constraint, and the whole process is repeated for a different local optimum γ_{k+1}.

5 APPLICATION OF THE FRAMEWORK

This section illustrates the proposed formalism on two distinct examples: a simplified practical environment, and a grid world. Both demonstrations have certain modelling properties relevant when developing solutions at scale. The former bridges the previous theoretical considerations with an applied perspective, and the latter dives deeper into computational machinery of the framework.

5.1 Tax optimization

Consider a transition system modelling transfer of an *intellectual property* (IP) among international companies, and transactions of royalty payments for the usage of the IP. Each agent (company) has tax residence as its main attribute, together with a variety of owned assets, e.g. IP ownership, account balance, company shares. Attributes of each agent define the state of the transition system. The actions allowing transitions are: `IP-transfer(oldOwner, newOwner)` to transfer IP, `IP-sublicense(licensor, licensee)` to sublicense IP, `payRoyalties(payer, receiver)` for paying for the use of the IP. Moreover, each company can create a new (child) company using the action `createChild(attributes)` (attributes are specified as a dictionary). For simplicity, we assume the system provides these actions without the need to specify further details.

5.1.1 Initial state and the utility. Let us define the initial state of the system consisting of a US-based company named *Multi-national enterprise*, which owns intellectual property `IP`, and is active in Portugal through a child company named *Portuguese company*. Clearly, the profits of the child company need to be transferred to its parent. The utility function can be thus defined as the profits minus the aggregate transaction costs. These costs predominantly depend on the tax code of the country of residence, namely US and Portugal. However, if the transactions are passing through different countries, then various tax benefits may apply. The goal of the agent is to find the most profitable transaction sequence, i.e. to find a sequence of actions minimizing the amount of taxes paid.

5.1.2 Optimal sequence. Recalling section 2, the optimal sequence of actions to minimize taxes paid in Portugal or US (prior to 2015) is to use Double Irish–Dutch Sandwich base erosion and profit shifting scheme [17]. In our case, this is implemented by writing 11 commands:

```
createChild({name: Irish company A, tax residence: Bermuda})
IP-transfer(Multi-national enterprise, Irish company A)
createChild({name: Dutch company, tax residence: Netherlands})
IP-sublicense(Multi-national enterprise, Dutch company)
createChild({name: Irish company B, tax residence: Ireland})
IP-sublicense(Dutch company, Irish company B)
IP-sublicense(Irish company B, Portuguese company)
payRoyalties(Portuguese company, Irish company B)
payRoyalties(Irish company B, Dutch company)
payRoyalties(Dutch company, Irish company A)
payRoyalties(Irish company A, Multi-national enterprise)
```

Following this scheme, the profits v are received in the form of royalty payments with little to no tax. The essence of the Double Irish–Dutch Sandwich can be described in a relatively concise way, attaining a form that can be regarded as minimal, because removing or inverting any action would label the scheme useless. A scheme ought to be therefore understood always as a set of non-compliant activity following a certain pattern, with a minimal form being present as the main representative of the set.

5.1.3 Complexity and camouflage. In practice, the Double Irish–Dutch Sandwich scheme is far more complex, as the network of child companies, IP sublicensing, or royalty payments is typically substantially larger. In fact, this additional level of complexity can be associated with camouflaging behavior, i.e. additional actions performed with the purpose to make it more difficult to track all transactions by the tax authorities. This illustrates the idea of structural neighborhood of a scheme discussed in sec. 4.2. The agent can add some additional actions that preserve the operating principle, i.e. being structurally close to the minimal form, e.g. creating additional layer of child companies in various countries. If we assume that any camouflage incurs a cost on its own, and then camouflaging will decrease the utility of non-compliance, then the minimal form is locally optimal. If a data-driven detection method is used to penalize the utility function, then the camouflaging costs are outweighed by the benefits of lower detection risks.

5.1.4 Abstraction. In order to formally describe the scheme as a set, one can use the minimal form representative to find its structural neighbors that are non-compliant. By perturbation of the minimal form located in the local optimum of the utility function, one can either generate a new instance of the scheme, or make the sequence of actions compliant. In the case of Double Irish–Dutch Sandwich scheme, a more complex instance can be created by inserting a few additional commands, e.g.:

```
createChild({name: Dummy company, tax residence: Tax heaven})
IP-sublicense(Irish company A, Dummy company)
payRoyalties(Dummy company, Irish company A)
```

anywhere into the sequence above. The issue is that there can be arbitrarily more complex instances, and not all of them might be necessarily non-compliant. For example, adding a command:

```
IP-sublicense(Multi-national enterprise, Portuguese company)
```

makes the sequence of transactions compliant, because the tax on royalty payments would be paid. For this reason, one needs to have a sufficiently general formal model and an efficient way to learn the description of the scheme, capturing all its instances.

[3]Although we assume that the expressions drawn from the grammar G are fully understandable by the human, it might still be the case that the expression is too complex to be analyzed in a reasonable time.

5.2 Pathfinding with hard and soft constraints

At this point, we illustrate the application of the proposed framework on a 8×8 grid environment displayed on Figure 2 with hard (blue tiles) and soft constraints (red tiles, unknown at start). In this environment, an agent is tasked with finding the sequence of state-action pairs that maximize the total reward as a sum of its immediate rewards. After every action, the agent gets immediate reward -0.1, except when reaching the end position with reward of 50. Since we assume we do not know where are the non-compliant positions, we cannot integrate this knowledge into the reward function at the start of the learning process.

The state of the system observed by the agent is its coordinate position. The set of actions is `up, left, right, down` movement from any position, except if the agent is about to get out of the grid or is about to violate the hard constraints, in which case the action is impossible to make. This environment exhibits two important properties that will simplify the computation:

- *objective accountability*: in order to decide if a sequence is in Γ, it is sufficient to consider the states of the system as partially observed by the agent.
- *Markov "normative" property*: a sequence is in Γ if and only if there exists a state $s \in \Gamma$ such that the transition sequence goes through s.[4]

The first property clearly holds, as the path of the agent is uniquely defined by the sequence of its own positions. This means that the environment can be modelled only as a sequence of positions, being the states of the transition system, and not necessarily as a sequence of states of the entire environment, which greatly simplifies the computation. The second property holds because every path over the positions violates the soft constraint if and only if at least one position of the path is inside the red region. By both properties, the information that will be recorded into $\mathcal{D}$ are the positions of the agent.

5.2.1 Utilitarian agent. The goal of the utilitarian agent is to reach the end position starting from the `start` position so that its utility is maximized. The choice of the algorithm to solve this task ought to be suited well for the given environment, as some algorithms can perform better in specific environments. In this study, we chose reinforcement learning, as this class of algorithms appear to generally scale well across various environments. We implement the utilitarian agent using standard Q-learning to find the optimal path [39], with discount factor $\omega = 0.9$, and learning rate $\alpha = 0.5$.[5] Starting from the `start` position, we perform 1000 episodes with 200 actions per episode during the training phase. The ϵ-decay is set to 0.001 with initial value 1.0. If the algorithm converges, the optimal path represents the local optimum γ_k.

5.2.2 Perturbation routine. After the optimal sequence is found, the perturbation routine is used to generate the neighborhood. We chose a very simple perturbation routine that will explore the

[4]Analogous to Markov property of memorylessness, i.e. the compliance status is independent of the path and depends only on individual states visited.
[5]Q-learning [39] approximates a Q-function $Q(s,a)$ capturing aggregated rewards by following an iterative procedure:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \cdot [R(s,a) + \omega \cdot max_{a'} Q(s',a') - Q(s,a)]$$

where $R(s,a)$ is the immediate reward, ω is the discount factor, α is the learning rate, and s' is the new state. This Q-function is then used to select actions of the agent.

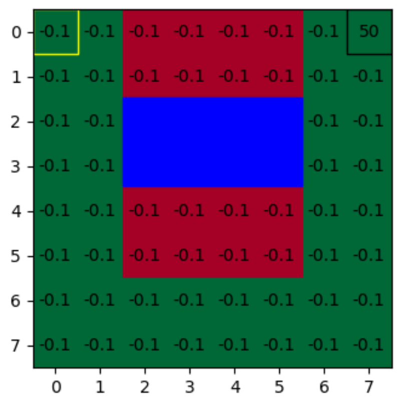


Figure 2: The grid environment with soft and hard constraints marked by red and blue, respectively. The `start` state of the agent is at position $(0,0)$ and the end state is position $(7,0)$. The immediate reward (cost) of each move action to a square is -0.1, except for the blue region that is not accessible by the agent, and the end position with reward equal to 50. Note that the red regions do not have higher reward, which means the agent has no intrinsic motivation to violate the soft constraint.

positions close to the positions traced by the optimal path. The agent starting on the `start` position and acting according to the optimal Q-table takes a random action with probability $p = 0.1$ to deviate from the optimal path. Since the agent is acting according to the Q-table, it will converge back to the optimal path in the next action to eventually reach the end position. Since the Markov normative property holds, one does not need to record the entire sequence, but only needs to record the positions that were not previously visited. Following this routine, the agent will eventually visit all positions in proximity to positions traced by the optimal path. In order to obtain a good sample of the neighborhood $\mathcal{N}(\gamma_k)$, we run this perturbation routine for 50 paths.

5.2.3 Active learning component. Initially, only the compliance status of the `start` position and the end position is known, and labeled with 1. As discussed in section 4, it is desirable to query instances with score close to zero. We model this requirement by defining an *acquisition function* f, that maps the scores $\mathbf{w}$ to a vector of the same length, such that $f(-1) = f(1) = 0$ and $\max_{x \in (-1,1)} f(x) = f(0)$. A function that satisfies these requirements is the kernel of a shifted Beta distribution (see Figure 3). Once the vector $\mathbf{w}$ is mapped by the acquisition function, the output vector is normalized in order to define a discrete probability distribution over the corresponding instances. Using this distribution, a random instance is queried. Once the label is obtained from the oracle, the score vector $\mathbf{w}$ is updated including the new label, and the labeling process is repeated until the budgetary constraint is met. Once the labeling process reaches its budgetary constraint, one more update of $\mathbf{w}$ is performed, and the final dataset is formed using the threshold rule $|w_i| > \eta = 0.65$ for each generated instance i.

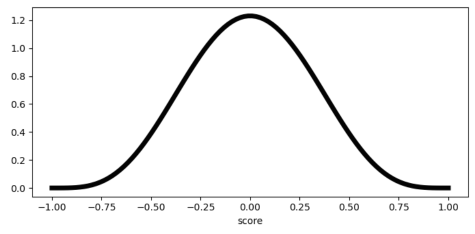


Figure 3: Kernel of a shifted Beta distribution with parameters 5, 5. The shape of the curve defines the strength of preference to exploration. Instances with score equal to -1 and 1 are mapped to zero, which means there is zero probability to be selected for a query. Conversely, instances with the score equal to zero have the highest chance to be picked for a query.

5.2.4 Semi-supervised learning component. Graph-based transduction learning can be used to obtain the score vector $\mathbf{w}$ relevant for query strategy [34]. In order to obtain the similarity matrix W, a similarity measure is used to calculate the similarity between every two instances. In our case, we use the Euclidean distance between the position coordinates. Once the similarity matrix W is calculated, the next step of the graph-based transduction learning is construction of the graph. The graph is constructed by identifying the instances with nodes, and the edge set E is formed by drawing an edge between all pairs of nodes with similarity lower than $\tau = 1.5$. Finally, the score vector $\mathbf{w}$ is obtained using an inference algorithm.

While several methods were developed [34], we chose the Gaussian random fields method [40] to find the labeling score $\mathbf{w}$, because of its simplicity and ability to calculate the score while keeping the score of known labeled instances equal to 1 or -1. The algorithm propagates known label information over the network, and assigns a score to each node as the weighted average of the score of the label assigned to its neighbor. The scores can be obtained iteratively by updating the score according to the following formula:

$$w_v \longleftarrow \frac{\sum_{(u,v\in E)} W_{uv} w_u}{\sum_{(u,v\in E)} W_{uv}}$$

for every unlabeled node v using labeled and unlabeled nodes u. The vector $\mathbf{w}$ is initialized using the true labels and random values for unknown labels. During the simulation, we have used 50 iteration updates.

Graph-based transduction admits a natural interpretation in the grid environment, as can be seen in the lower row on Figures 4. The score of every square is influenced by its closest neighboring squares, and the parameter τ controls the maximal distance of influence. The Gaussian random field algorithm can then be interpreted as propagation of known label information through the grid, with τ controlling the distance of the information source.

5.2.5 Automated formalization. In section 1, we claimed that non-compliant behavior has the tendency to cluster into schemes, i.e. classes in the space Γ. This is a core requirement not only for the semi-supervised learning component, but also for the design of the grammar. In the grid example, this property holds, and manifests by the non-compliant (red) regions forming connected areas. This means one can consider a grammar that is formalizing the notion of an area, that is, a geometric object expressed in a mathematical language. In this illustrative example, we already know a rectangle is the most suitable. Assuming this type of regularity, the high level representation covering the set Γ_k for $k = 1, 2$ is defined by the expression $\{(x, y)|x_1 \leq x \leq x_2, y_1 \leq y \leq y_2\}$, interpreted in the standard geometric way as all position coordinates contained in a square defined by two opposite corner points (x_1, y_1) and (x_2, y_2). Even after correctly identifying the most suitable description for the subsets of Γ, solving the optimization problem (2) is still a non-trivial task. A heuristic method is employed to find optimal (x_1, y_1) and (x_2, y_2) by randomly sampling points from the grid. To restrict the expressibility of the high-level expression, that is, the covering area of the rectangle, we consider a penalty of the objective function. The value of the objective function is multiplied by $(ab)^{-h}$, where a and b are the length and height of the rectangle. This means that the objective function is penalized by the area of the proposed rectangle controlled by the parameter h, thus restraining the solution from covering too many unlabeled data. We set this parameter to 0.25.

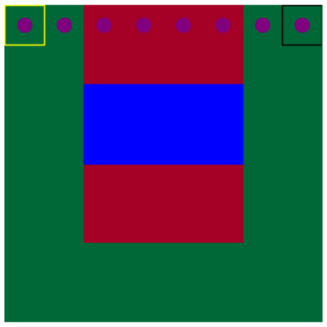
5.2.6 Simulation results. On Figure 4 we can see how the framework managed to include most of the set $\Gamma = \Gamma_1 \cup \Gamma_2$ as hard constraints. The sensitivity of the parameters of the relevant algorithms remains to be investigated, but generally it appeared that the framework provides a good robustness, and manages to cover a significant part of Γ within 3 iterations with the budget of 5 queries per iteration. Note that even if there are still some remaining red squares in the environment (eg, Figure 4c), the paths leading through the remaining red squares are no longer profitable. This means there is no reason for the utilitarian agent to cross the red squares, and can choose a compliant path of higher utility. We also observed that the component mainly influencing the success of the framework is the quality of the query. If less informative squares happen to be sampled, then the framework can fail in blocking all profitable non-compliant paths.

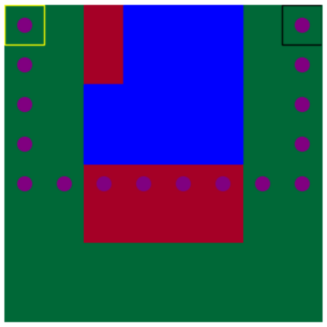
6 DISCUSSION

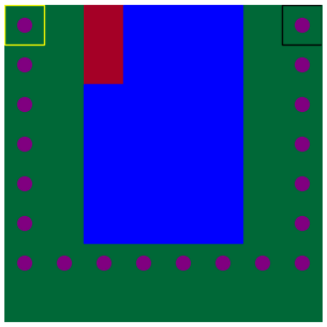
The strength of the proposed framework lies in its modularity. Every component of the system, whether it is the algorithm for solving the optimization task (1), the perturbation method, the query strategy, label inference, or the grammar and search algorithm for the optimization task (2), can be substituted by a tool more suitable for the given context.

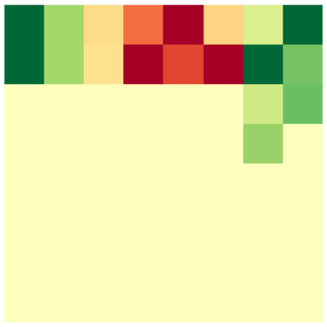
6.1 Formal model choice

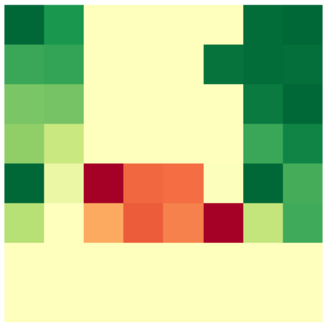
In the example in section 5.2 the grammar used consisted only of the expression of a rectangular area with two free variables. If the non-compliant areas were circles or any other shapes, then the set Γ would be either insufficiently covered, or covered including many compliant instances. Having no prior knowledge on the topology of Γ makes it difficult to choose a suitable model. The grammar could be extended with different geometric objects, but this would come at the cost of a larger search space. In the example in section 5.1, the task gets more difficult due to the fact that the minimal form of the Double Irish–Dutch Sandwich scheme can be extended with an


(a) The environment with all non-compliant profitable paths free.


(b) The first soft constraint is integrated, the agent finds a new optimal path.


(c) Integration of the last region results in blockage of all profitable paths.


(d) Inferred information about the optimal neighborhood


(e) Inferred information about the new optimal neighborhood.

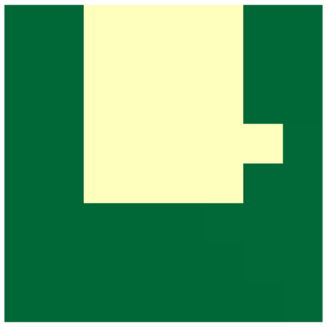

(f) The set $\mathcal{D}_{-1}$ is empty, the inference process labels all instances with 1.

Figure 4: In the upper row, we see the optimal paths of the agent. In (a), initially, all soft constraints (red area) are crossable. The agent finds the optimal path γ_1 (marked in purple dots). Using the optimal path, the perturbation routine generates the neighborhood $\mathcal{N}(\gamma_1)$ and the oracle performs labeling with the help of graph-based transduction learning. In (d) the score vector w of the neighborhood is plotted, where green squares indicate high certainty of a compliant position and the red squares indicate high certainty of non-compliant position. The squares with yellowish color indicate uncertainty about the compliance status of the square. All instances with score higher than the threshold $\eta = 0.65$ are included into the automated formalization process. In (b), we can see that the system identified the norm-violating region, which means this region is integrated as a hard constraint. In the next iteration step, the agent is free to find a new optimal path γ_1. The formalization process is repeated with a new score vector w, as visible in (e), and another norm-violating region is integrated. The agent is free again to find a new path, however, as can be observed in (f) there are no known norm-violating instances; the completion process terminates.

arbitrary amount of camouflaging actions. These are examples of insights relevant for the grammar design.

6.2 Similarity and multi-agent extension

While research on autonomous utilitarian agents have progressed rapidly, the main weakness of the framework appears to be its reliance on the notion of similarity to extrapolate knowledge from a few labeled instances. General similarity functions cannot be expected to reliably extract relevant information about the true similarity between legal cases. Metric learning might however mitigate the issue [20]. A more crucial limitation of the preset framework is that it is not applicable in multi-agent settings (cooperating, competing). Recent advancements in the area of (utilitarian) multi-agent learning seems to be promising in solving the optimization problem (1) [21]. However, the identification of the cooperative behavioral pattern into an explainable formal expression remains a most prominent challenge in the area of norm identification [6], and in principle our framework could provide a starting point.

7 CONCLUSIONS

Norm identification, induction, formalization, and integration, is still an open topic in the area of AI & law. While the interaction between an autonomous agent and a human providing sparse feedback is a traditional topic in robotics, our study brings forth this question in the context of normative systems and compliance checking.

From the point of view of a potential perpetrator, formalization of laws can also be used as an efficient tool for optimization of norm-violating behavior. The framework proposed in this study has the potential to act as an effective countermeasure. For their modularity and generality, the principles outlined in our study may serve as methodological guidelines for developing solutions at scale. On the experimental side, further research is needed on applying our or similar frameworks on use-cases of compliance checking systems used in practice, to compare and benchmark various available state-of-art AI methods, possibly extending the framework to a multi-agent setting.

ACKNOWLEDGMENTS

This work was partly funded by the Dutch Research Council (NWO) for the HUMAINER AI project (KIVI.2019.006).

REFERENCES

[1] David W Aha. 1991. Case-based learning algorithms. In *Proceedings of the 1991 DARPA case-based reasoning workshop*, Vol. 1. 147–158.

[2] Robert Amor and Johannes Dimyadi. 2021. The promise of automated compliance checking. *Developments in the built environment* 5 (2021), 100039.

[3] Kevin D. Ashley. 1992. Case-Based Reasoning and its Implications for Legal Expert Systems. *Artificial Intelligence and Law* 1, 2-3 (1992), 113–208.

[4] Massimo Bartoletti, Salvatore Carta, Tiziana Cimoli, and Roberto Saia. 2020. Dissecting Ponzi schemes on Ethereum: identification, analysis, and impact. *Future Generation Computer Systems* 102 (2020), 259–277.

[5] Michaela Beals, Marguerite DeLiema, and Martha Deevy. 2015. Framework for a taxonomy of fraud. *Financial Fraud Research Center* (2015).

[6] Rafael C Cardoso, Brian Logan, Felipe Meneguzzi, Nir Oren, and Bruno Yun. 2022. Resilience, reliability, and coordination in autonomous multi-agent systems. *AI Communications* Preprint (2022), 1–18.

[7] Domenico Corapi, Alessandra Russo, Marina De Vos, Julian Padget, and Ken Satoh. 2011. Normative design using inductive learning. *Theory and Practice of Logic Programming* 11, 4-5 (2011), 783–799.

[8] Stephen Cranefield and Ashish Dhiman. 2021. Identifying Norms from Observation Using MCMC Sampling.. In *IJCAI*. 118–124.

[9] Stephen Cranefield, Felipe Meneguzzi, Nir Oren, and Bastin Tony Roy Savarimuthu. 2016. A Bayesian approach to norm identification. *ECAI 2016* (2016).

[10] Stephen Cranefield and Bastin Tony Roy Savarimuthu. 2021. Normative Multi-Agent Systems and Human-Robot Interaction. In *Workshop on Robot Behavior Adaptation to Human Social Norms (TSAR)*.

[11] Marina De Vos, Sabrina Kirrane, Julian Padget, and Ken Satoh. 2019. ODRL policy modelling and compliance checking. In *International Joint Conference on Rules and Reasoning*. Springer, 36–51.

[12] Peter Fratrič, Giovanni Sileno, Tom van Engers, and Sander Klous. 2022. Computational Discovery of Transaction-Based Financial Crime via Grammatical Evolution: The Case of Ponzi Schemes. In *Coordination, Organizations, Institutions, Norms, and Ethics for Governance of Multi-Agent Systems XV: International Workshop, COINE 2022, Virtual Event, May 9, 2022, Revised Selected Papers*. Springer, 109–120.

[13] Javier Garcia-Bernardo, Jan Fichtner, Frank W Takes, and Eelke M Heemskerk. 2017. Uncovering offshore financial centers: Conduits and sinks in the global corporate ownership network. *Scientific reports* 7, 1 (2017), 1–10.

[14] Mustafa Hashmi, Guido Governatori, Ho Pun Lam, and Moe Thandar Wynn. 2018. Are we done with business process compliance: state of the art and challenges ahead. *Knowledge and Information Systems* 57 (10 2018), 79–133. Issue 1. https://doi.org/10.1007/s10115-017-1142-1

[15] Erik Hemberg, Jacob Rosen, Geoff Warner, Sanith Wijesinghe, and Una-May O'Reilly. 2016. Detecting tax evasion: a co-evolutionary approach. *Artificial Intelligence and Law* 24 (2016), 149–182.

[16] Yu-Chi Ho and Xi-Ren Cao. 1991. *Perturbation Analysis of Discrete Event Dynamic Systems*. Springer US. https://doi.org/10.1007/978-1-4615-4024-3

[17] Chris Jones, Yama Temouri, and Alex Cobham. 2018. Tax haven networks and the role of the Big 4 accountancy firms. *Journal of World Business* 53, 2 (2018), 177–193. https://doi.org/10.1016/j.jwb.2017.10.004

[18] Robert Kowalski and Akber Datoo. 2021. Logical English meets legal English for swaps and derivatives. *Artificial Intelligence and Law* (2021), 1–35.

[19] Naeimeh Laleh and Mohammad Abdollahi Azgomi. 2009. A taxonomy of frauds and fraud detection techniques. In *Information Systems, Technology and Management: Third International Conference, ICISTM 2009, Ghaziabad, India, March 12-13, 2009. Proceedings 3*. Springer, 256–267.

[20] Dewei Li and Yingjie Tian. 2018. Survey and experimental study on metric learning methods. *Neural Networks* 105 (9 2018), 447–462. https://doi.org/10.1016/j.neunet.2018.06.003

[21] Siqi Liu, Guy Lever, Zhe Wang, Josh Merel, SM Ali Eslami, Daniel Hennes, Wojciech M Czarnecki, Yuval Tassa, Shayegan Omidshafiei, Abbas Abdolmaleki, et al. 2022. From motor control to team play in simulated humanoid football. *Science Robotics* 7, 69 (2022), eabo0235.

[22] Bertram F Malle, Eric Rosen, Vivienne B Chi, Matthew Berg, and Peter Haas. 2020. A general methodology for teaching norms to social robots. In *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 1395–1402.

[23] Paulus Merks. 2006. Tax Evasion, Tax Avoidance and Tax Planning. *Intertax* (2006), 272–281.

[24] Kevin Miller, Jack Mauro, Jason Setiadi, Xoaquin Baca, Zhan Shi, Jeff Calder, and Andrea L Bertozzi. 2022. Graph-based active learning for semi-supervised classification of SAR data. In *Algorithms for Synthetic Aperture Radar Imagery XXIX*, Vol. 12095. SPIE, 126–139.

[25] Cyril Onwubiko. 2020. Fraud matrix: A morphological and analysis-based classification and taxonomy of fraud. *Computers & Security* 96 (2020), 101900.

[26] Monica Palmirani, Guido Governatori, Antonino Rotolo, Said Tabet, Harold Boley, and Adrian Paschke. 2011. LegalRuleML: XML-Based Rules and Norms. In *Rule-Based Modeling and Computing on the Semantic Web*. Springer Berlin Heidelberg, Berlin, Heidelberg, 298–312.

[27] Edwina L Rissland, Kevin D Ashley, and L Karl Branting. 2005. Case-based reasoning and law. *The Knowledge Engineering Review* 20, 3 (2005), 293–298.

[28] Vasanth Sarathy, Matthias Scheutz, and Bertram F Malle. 2017. Learning behavioral norms in uncertain and changing contexts. In *2017 8th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*. IEEE, 000301–000306.

[29] Bastin Tony Roy Savarimuthu, Stephen Cranefield, Maryam A Purvis, and Martin K Purvis. 2010. Obligation norm identification in agent societies. *Journal of Artificial Societies and Social Simulation* 13, 4 (2010), 3.

[30] Burr Settles. 2011. From theories to queries: Active learning in practice. In *Active learning and experimental design workshop in conjunction with AISTATS 2010*. JMLR Workshop and Conference Proceedings, 1–18.

[31] Sepehr Sharifi, Alireza Parvizimosaed, Daniel Amyot, Luigi Logrippo, and John Mylopoulos. 2020. Symboleo: Towards a specification language for legal contracts. In *2020 IEEE 28th International Requirements Engineering Conference (RE)*. IEEE, 364–369.

[32] Michael Siering, Benjamin Clapham, Oliver Engel, and Peter Gomber. 2017. A taxonomy of financial market manipulations: establishing trust and market integrity in the financialized economy through automated fraud detection. *Journal of Information Technology* 32, 3 (2017), 251–269.

[33] Giovanni Sileno, Alexander Boer, and Tom van Engers. 2017. Reading agendas between the lines, an exercise. *Artificial Intelligence and Law* 25 (2017), 89–106.

[34] Amarnag Subramanya and Partha Pratim Talukdar. 2014. *Graph-Based Semi-Supervised Learning*. Springer International Publishing. https://doi.org/10.1007/978-3-031-01571-7

[35] Zhi-Xuan Tan, Jake Brawer, and Brian Scassellati. 2019. That's mine! learning ownership relations and norms for robots. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 8058–8065.

[36] L Thomas Van Binsbergen, Lu-Chi Liu, Robert van Doesburg, and Tom van Engers. 2020. eFLINT: a domain-specific language for executable norm specifications. In *Proceedings of the 19th ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences*. 124–136.

[37] Jesper E. van Engelen and Holger H. Hoos. 2020. A survey on semi-supervised learning. *Machine Learning* 109 (2 2020), 373–440. Issue 2. https://doi.org/10.1007/s10994-019-05855-6

[38] Vladimir N. Vapnik. 2000. *The Nature of Statistical Learning Theory*. Springer New York. https://doi.org/10.1007/978-1-4757-3264-1

[39] Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning* 8 (1992), 279–292.

[40] Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. 2003. Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning* (Washington, DC, USA) *(ICML'03)*. AAAI Press, 912–919.