Journal of Artificial Intelligence Research 75 (2022) 1549-1593 Submitted 02/2022; published 12/2022

Data-Driven Revision of Conditional Norms in Multi-Agent Systems

Davide Dell'Anna D.DELLANNA@TUDELFT.NL
Delft University of Technology, Delft, The Netherlands

Natasha Alechina N.A.ALECHINA@UU.NL
Fabiano Dalpiaz F.DALPIAZ@UU.NL
Mehdi Dastani M.M.DASTANI@UU.NL
Utrecht University, Utrecht, The Netherlands

Brian Logan B.S.LOGAN@UU.NL
Utrecht University, Utrecht, The Netherlands
University of Aberdeen, Aberdeen, United Kingdom

Abstract

In multi-agent systems, norm enforcement is a mechanism for steering the behavior of individual agents in order to achieve desired system-level objectives. Due to the dynamics of multi-agent systems, however, it is hard to design norms that guarantee the achievement of the objectives in every operating context. Also, these objectives may change over time, thereby making previously defined norms ineffective. In this paper, we investigate the use of system execution data to automatically synthesise and revise conditional prohibitions with deadlines, a type of norms aimed at prohibiting agents from exhibiting certain patterns of behaviors. We propose DDNR (Data-Driven Norm Revision), a data-driven approach to norm revision that synthesises revised norms with respect to a data set of traces describing the behavior of the agents in the system. We evaluate DDNR using a state-of-the-art, off-the-shelf urban traffic simulator. The results show that DDNR synthesises revised norms that are significantly more accurate than the original norms in distinguishing adequate and inadequate behaviors for the achievement of the system-level objectives.

1. Introduction

Multi-agent systems (MASs) comprise autonomous agents that interact in a shared environment (Wooldridge, 2009). For example, a smart traffic system is a MAS that includes autonomous agents like cars, pedestrians, smart traffic lights, etc. To achieve the system-level objectives of a MAS, the behavior of the autonomous agents should be controlled and coordinated (Bulling and Dastani, 2016). Norms and their enforcement have been proposed as a way to control and coordinate the behavior of the agents in a MAS without limiting their autonomy (Chopra et al., 2018; Tinnemeier et al., 2009; Testerink et al., 2016). Similar to our society, norms can be viewed as standards of behavior specifying that certain states or sequences of actions in a MAS should occur (*obligations*) or should not occur (*prohibitions*) in order for the objectives of the MAS to be achieved (Boella and van der Torre, 2004).

We investigate the following scenario of MAS engineering. A *MAS designer* defines the overall structure of a MAS by specifying a set of norms that the agents who participate in the MAS are expected to obey. One or more *agent designer*s, possibly one per agent and each different from the MAS designer, determine the internal architecture of the agents.

For many applications, it is assumed that the behavior of the agents and the norms enforced in the MAS will guarantee the achievement of the system's objectives (Dastani et al., 2009). This is possible, for example, when the MAS designer and agent designers are the same people, or when the agent's behavior is enforced via the regimentation of norms, so that the agents cannot deviate from the intended behavior.

Regimentation is particularly difficult in *open MASs* (Artikis and Pitt, 2001), where agents can freely enter or leave the system, and their internal architecture is not known to the MAS designer. Regimenting norms in open MASs has undesirable side effects: while it can ensure that the objectives are achieved (if the norms are aligned with the objectives), agent autonomy is unnecessarily restricted. Furthermore, without awareness of how the agents internally behave, the MAS designer cannot fully predict how the agents will react to the norms, making it impossible, or computationally infeasible, to design norms that guarantee the satisfaction of the system's objectives in every possible combination of the agents' behaviors. Also, the MAS objectives may change over time, and the designed norms may become outdated and ineffective for the new objectives (Bicchieri, 2005).

To cope with these issues, norms need to be continuously evaluated, and possibly revised when they become inadequate for achieving the MAS objectives (Dell'Anna et al., 2020). Several formal contributions to norm revision have been proposed, including logics for norm change (Aucher et al., 2009; Knobbout et al., 2016), design patterns for the iterative revision and verification of a specification (Kafali et al., 2017), or the automated refinement of normative specifications via inductive logic programming (Corapi et al., 2011). Most of the literature, however, assumes that the MAS designer possesses explicit knowledge of the agents' internals or of the causal relationship between agent behavior, norms and MAS objectives. For example, our previous work (Dell'Anna et al., 2020) proposed a mechanism for revising the sanctions that are used to enforce norms. However, we assume knowledge of the preferences of the agents in the system. In (Corapi et al., 2011), inductive logic programming is used, assuming that norms and objectives are expressed in the same language. The objectives of MASs, however, are often stated in a language with no direct connection with the language for expressing norms. In a traffic system, for example, one objective may be to avoid traffic collisions, but 'not colliding' is not a property under full agent control, and prohibition of collisions cannot be stated as a norm, as this also depends on other agents and on the weather conditions, among other factors.

The increasing availability of large amounts of system behavior data (van der Aalst, 2016; Lorenz et al., 2021) enables new approaches for the automated design of norms that relax these assumptions and where the revision of norms starts from data collected during the execution of the system. For example, data may show that collisions always happen when an agent's speed is high, allowing us to state a speed limiting norm that prohibits agents from driving faster than 100 km/h.

Based on the discussion above, this paper makes the following design decisions to support norm revision in open and dynamic MASs: (1) we choose for the exogenous control of a MAS, i.e., we do not assume control over agents' design; (2) we study how to synthesise and revise norms from system execution data describing the behavior of the agents in a MAS *at run-time*, thereby supporting the dynamics within a MAS as well as different specification languages for MAS objectives and norms; (3) we propose a norm revision mechanism that does not require regimentation of norms, but supports enforcement via sanctioning that respects better the autonomy of the agents (Alechina et al., 2013).

Specifically, in this paper we make the following contributions:

- We introduce *DDNR*, a novel Data-Driven Norm Revision approach. *DDNR* consists of two steps: the *synthesis step* generates candidate revisions from a set of norms; the *selection step* chooses the final revised norms from the set of candidate revisions based on how well they characterize the available data. We focus on the revision of conditional norms (prohibitions) with deadlines, a type of norms used in the normative MAS literature to express behavioral properties (Tinnemeier et al., 2009).
- We report on the complexity of the proposed Data-Driven Norm Revision approach.
- We apply and experimentally evaluate a Java implementation of *DDNR*, available at (Dell'Anna et al., 2022a), on an agent-based traffic simulation where norms regulate the behavior (maximum speed and minimum safety distance) of vehicles on a highway.

Organization. Section 2 provides the necessary background on (normative) MASs and conditional norms. Section 3 describes the proposed Data-Driven Norm Revision (*DDNR*) approach and its complexity. Section 4 reports on our empirical evaluation in the context of agent-based traffic simulation. Section 5 discusses the limitations. Section 6 presents related work, and Section 7 outlines conclusions and future work.

2. Norm Revision in Normative Multi-Agent Systems

The focus of this paper is on Normative Multi-Agent Systems (NMASs), i.e., MASs where norms are enforced in order to achieve system-level objectives. These objectives characterize the desired behavior of either the MAS or of the agents. Our goal is to devise an automated and data-driven mechanism for norm revision, which ensures that the enforced norms contribute to the achievement of the MAS objectives.

As an illustrative example of a NMAS, we use the highway section shown in Figure 1. In line with the idea of open MASs, we assume that the internals of the agents in the MAS (viz., the vehicles driving on the highway) are not known to the MAS designer: the agents are treated as black boxes. We assume, however, that the MAS is endowed with a Monitoring and Norm Enforcement component that communicates to the agents the enforced norms and the consequences of violating them (e.g., the sanctions), so that the agents entering the system are norm-aware and can take autonomous decisions accordingly.

We focus on conditional norms with deadlines, which express behavioral properties (Tinnemeier et al., 2009). In our example, we consider norms concerning the speed limit of the vehicles and their minimum safety distance. For instance: *if a vehicle enters the 2nd km of the highway, it is prohibited from driving faster than 70 km/h until it reaches the 7th km of the highway.* Conditional norms are detached in certain states of the MAS (e.g., the vehicle entering the 2nd km of the highway) and have a temporal validity specified by a deadline (the vehicle reaches the 7th km). The satisfaction or violation of a detached norm depend on whether the behavior of the agent brings about a prohibited state (speed >70 km/h) before a state in which the deadline condition becomes true.

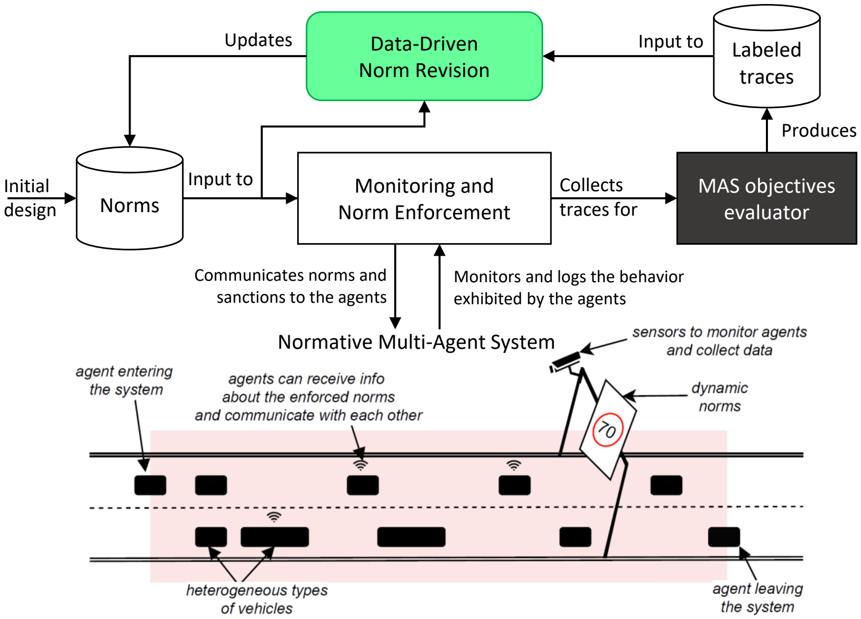

Updates
Data-Driven Norm Revision
Input to
Labeled traces
Produces
Initial design
Norms
Input to
Monitoring and Norm Enforcement
Collects traces for
MAS objectives evaluator
Communicates norms and sanctions to the agents
Monitors and logs the behavior exhibited by the agents
Normative Multi-Agent System
sensors to monitor agents and collect data
agent entering the system
agents can receive info about the enforced norms and communicate with each other
dynamic norms
70
heterogeneous types of vehicles
agent leaving the system

Figure 1: A Normative Multi-Agent System, where norms are used to control the behavior of the autonomous agents (small black rectangles resembling vehicles) in the MAS. The Data-Driven Norm Revision module revises the norms based on the collected data labeled by the MAS objectives evaluator (labeled traces). The MAS objectives evaluator provides a labeling of the monitored agents' behaviors (collected traces) w.r.t. the MAS objectives.

The Monitoring and Norm Enforcement component monitors agent behavior, and stores the collected data in a database in the form of a *data set of finite traces*. Finite traces are collected via runtime monitoring of the MAS (Alechina et al., 2014). A *trace* in the running example represents the car journey through the highway, and it is generated by the actions of a vehicle. The collected traces are then evaluated by the MAS objectives evaluator component, which labels each trace as either *positive* or *negative* with respect to the MAS objectives. We assume that each trace describing the behavior of the agents can be labeled as either positive or negative based on whether it contributes positively to the achievement of the MAS objectives, but we do not make assumptions on how such labeling is obtained[1]. The MAS objectives evaluator can therefore be seen as an external component, either human or automated, beyond the scope of the Data-Driven Norm Revision module and of this paper.

This paper focuses on the Data-Driven Norm Revision module (the green box in Figure 1), which revises the currently enforced norms so to ensure that (i) traces (behaviors) that are

[1] This assumption is realistic in several contexts and different kinds of MAS objectives (e.g., instances of a process can be deemed as compliant or non-compliant w.r.t. a model (Loreti et al., 2020); in our traffic example, traces can be labeled individually w.r.t. their travel time or emissions). Since we do not make assumptions about the source of labeling, however, we also cover cases where the MAS objectives depend on the *joint* behavior of different agents (e.g., the throughput of a road) and labels are assigned to *groups of traces* (e.g., all traces collected in the time period when the throughput was below desired levels are labeled as negative). We focus here on the case in which every trace is labeled independently, and leave other scenarios for future work.

labeled as negative by the MAS objectives evaluator are prohibited by the revised norms, and (ii) traces that are labeled as positive by the MAS objectives evaluator are not prohibited by the revised norms. The Data-Driven Norm Revision module is *agnostic to the MAS objectives*, for it only relies on labeled traces provided by the MAS objectives evaluator, which are considered as ground truth. This guarantees that the proposed Data-Driven Norm Revision module is data driven and supports those cases where the MAS objectives do not correspond directly to properties expressible in the language of the norms (e.g., we can express norms concerning the speed of the vehicles, while the objectives may concern CO_2 emissions) or where the causal relationship between norms and objective is not known or unavailable. In our experiments that implement the running example (Section 4), we label the traces that represent the cars journey through the highway section based on a combination of measurements of the CO_2 levels that were emitted by the vehicle in the highway and its travel time. Neither the vehicles (agents) nor the Data-Driven Revision module will be provided information about such an objective.

2.1 Agent Behavior and Conditional Norms

We provide a formal definition of the basic concepts for the rest of the paper.

2.1.1 Language of States, Traces and Norms

We assume a finite propositional language $L_{\{\wedge,\vee\}}$ (simply L, from now on), with $\wedge$ and $\vee$ indicating the conjunction and disjunction propositional connectives, respectively. Propositions in L correspond to properties of states of the MAS. A *state* of the MAS, represented as s, is a (possibly empty) set of propositional variables (atomic propositions) considered true in that state. When describing a state, we use lower case letters (e.g., p, q), with the exception of the letter s, to indicate propositions. A *trace*, represented generically as γ, is a finite sequence of states. We use the notation $(s_1, \ldots, s_k)$ for a trace γ consisting of k states $s_1, \ldots, s_k$, where s_i indicates the *i-th* state in γ. We denote with $\mathcal{S}(\Gamma)$ or simply $\mathcal{S}$ the set of states occurring in traces in a data set of traces Γ.

In the running example, we consider a set of propositional variables $V = VP \cup VS \cup VT$, where $VP = \{km_1, \ldots, km_{10}\}$ is a subset of the variables referring to vehicle positions on the highway of the running example (i.e., km_i indicates that the vehicle reached the i^{th} km of the highway), $VS = \{sp_x \mid 10 \leq x \leq 140 \;\&\; x \in \mathbb{N}\}$ are propositions denoting that the vehicle's speed is *higher than* a certain speed x in *km/h*, and $VT = \{truck, car\}$ represents vehicle types. An example of a trace γ composed of 10 states and describing the behavior of a car in the running example is the following.

$$\gamma = (\{km_1, sp_{30}, car\}, \{km_2, sp_{22}, car\}, \{km_3, sp_{10}, car\}, \{km_4, sp_{32}, car\}, \{km_5, sp_{10}, car\}, \{km_6, sp_{18}, car\}, \{km_7, sp_{32}, car\}, \{km_8, sp_{10}, car\}, \{km_9, sp_{18}, car\}, \{km_{10}, sp_{14}, car\}) \quad (1)$$

The states in the trace in Equation 1 are indicated as sets containing the propositions that are true in that state. Furthermore, for brevity, here and in the rest of the paper we only report, for each state, the proposition from VS that represents the highest detected speed of the vehicle, and we omit (but do not ignore) all the other propositions that refer to the speeds that are lower than the detected speed, which are also true in the state due to their semantics, i.e., if the highest detected speed of a vehicle is v, all propositions $sp_x \in VS$ such that $x \leq v$ are true in the state. The same simplification of notation is also applied for propositions from VP. For example, let the set of propositions $\{km_1, km_2, car\} \cup \{sp_x \mid 10 \leq x \leq 22\}$ be called V_{true}. In the full representation of state s_2 in Equation 1, all propositions in V_{true} are true, and all propositions in $V \setminus V_{true}$ (i.e., propositions in $\{km_x \mid 2 < x \leq 10\} \cup \{sp_x \mid 22 < x \leq 140\} \cup \{truck\}$) are false.

2.1.2 Data Set and Classification of Traces

We consider a data set (i.e., a set) of traces Γ to be partitioned into two sets Γ_P (positive traces) and Γ_N (negative traces). The partition is performed by the MAS objectives evaluator described above. Column *Data set* Γ in Table 1 shows a simple example of Γ where $\Gamma_P = \{\gamma_1, \gamma_3\}$ and $\Gamma_N = \{\gamma_2, \gamma_4\}$.

We say that a conditional norm n *classifies* a trace from Γ as *norm-violating* (or simply *violating*) if the trace violates n, and as *norm-compliant* (or simply *compliant*) if the trace does not violate n. In other words, we interpret a norm as a binary classifier that distinguishes two types of traces: violating and compliant ones. A formal account on violation and compliance is provided later in Definition 2.

Therefore, we have two separate classifications of the traces in Γ: one is done by the MAS objectives evaluator (column Label in Table 1), and the other is done by the norms (e.g., columns *Norm n* and *Norm n'*). Since we are interested in ensuring that traces classified as negative by the MAS objectives evaluator are classified as violating by the norms, and traces that are classified as positive by the MAS objectives evaluator are classified as compliant by the norms, we can distinguish four different types of traces, like those reported in column *Type of trace w.r.t. label and n* of Table 1.

Table 1: Example of data set Γ, where traces are labeled by the MAS objectives evaluator (column Label), and of the classification of such traces as compliant or violating according to two different norms n and n'. Columns *Type of trace* show that norm n is not perfectly aligned with the MAS objectives, while norm n' is, because the traces are either true positives or true negatives.

Data set Γ		*Norm*	*Type of trace*	*Norm*	*Type of trace*
Trace	Label	n	w.r.t. label and n	n'	w.r.t. label and n'
γ_1	positive	compliant	True Positive	compliant	True Positive
γ_2	negative	compliant	False Positive	violating	True Negative
γ_3	positive	violating	False Negative	compliant	True Positive
γ_4	negative	violating	True Negative	violating	True Negative

- Trace γ_1 describes a behavior (trace) that is positive according to the given labeling and is also compliant with norm n. Borrowing terminology from statistics, we call γ_1 a True Positive w.r.t. n, to indicate that the trace is correctly (w.r.t. the MAS objectives evaluation) classified as norm-compliant by n.

- Trace γ_2 describes a behavior that is compliant with norm n (it does not violate n) but that should be prohibited, since γ_2 is labeled as negative. We call γ_2 a False Positive, to indicate that the trace is erroneously classified as norm-compliant by n.
- Trace γ_3 describes a behavior that currently violates n but that should not be prohibited, since γ_3 is labeled as positive. We say that γ_3 is a False Negative because the trace is erroneously prohibited (i.e., classified as norm-violating) by n.
- Finally, γ_4 describes a behavior that is currently prohibited by n and is labeled as negative. We call γ_4 a True Negative, for the trace is correctly prohibited by n.

Note that norm n in Table 1 correctly regulates only two of the four traces in Γ: it correctly classifies γ_1 as norm-compliant and γ_4 as norm-violating, but it erroneously classifies γ_2 as norm-compliant and γ_3 as norm-violating. Given a norm n that erroneously regulates some traces, our goal is to synthesise, using propositions from L, a *revised norm* that classifies as norm-compliant (more) positive traces and as norm-violating (more) negative traces in Γ. Norm n' in Table 1 provides an example of a norm that *correctly classifies* all traces w.r.t. their labeling, i.e., a norm that leads to no False Positive/Negative. We say that a norm like n' is *perfectly aligned with the MAS objectives.*

This concept can also be described by means of a confusion matrix (see Figure 2a). A confusion matrix describes the relationship between the classification of traces according to a norm (i.e., whether the trace is classified as norm-compliant or norm-violating) and the correct classification of the traces according to the MAS objectives labeling. Each cell (i, j) in the matrix contains the number of traces in the data set Γ that are classified as i by the MAS objectives evaluator and as j by the norm. For example, cell $(\textit{positive}, \textit{compliant})$ contains the number of traces in Γ that are labeled as positive w.r.t. the MAS objectives and classified as norm-compliant w.r.t. a norm, i.e., it contains the number of true positives (TP). The inner diagonal of the matrix (the diagonal from TP to TN) represents the number of traces correctly classified by a norm. The outer diagonal, instead, represents the number of errors, or misclassifications.

$$\begin{array}{cc|cc|} & & \multicolumn{2}{c}{\text{norm}} \\ & & \text{compliant} & \text{violating} \\ \text{objectives} & \text{positive} & \text{TP} & \text{FN} \\ & \text{negative} & \text{FP} & \text{TN} \end{array}$$

(a)

$$\begin{array}{cc|cc|} & & \multicolumn{2}{c}{\text{norm}} \\ & & \text{compliant} & \text{violating} \\ \text{objectives} & \text{positive} & x & 0 \\ & \text{negative} & 0 & |\Gamma| - x \end{array}$$

(b)

Figure 2: A generic confusion matrix (a) and an example with no misclassifications (b).

Given a data set of labeled traces Γ, the confusion matrix provides a compact representation of how well a norm is aligned with the MAS objectives. A perfectly aligned norm w.r.t. Γ implies the confusion matrix in Figure 2b, where all traces are correctly classified and no trace is misclassified.

Previous work has shown that verifying the existence of such a norm that is perfectly aligned with the MAS objectives w.r.t. the traces in Γ, like the norm n' in Table 1, is an NP-complete problem (Dell'Anna et al., 2022b). Motivated by this complexity result, we propose a polynomial heuristic algorithm to solve this problem in an approximate way. In particular, instead of searching for revised norms that are perfectly aligned with the MAS objectives, we search for revised norms that are *better aligned* with the MAS objectives, i.e., that lead to fewer False Positives/Negatives, than the original norms. More precisely, we define the *alignment* of a norm with the MAS objectives w.r.t. a data set of traces as the *accuracy*[2] of the norm in classifying the traces. More details are given in Section 3.3.

2.1.3 Conditional Prohibitions

We formally define conditional prohibitions and the conditions for the violation of conditional prohibitions. In the rest of the paper, we use the terms *norm* and *prohibition* interchangeably.

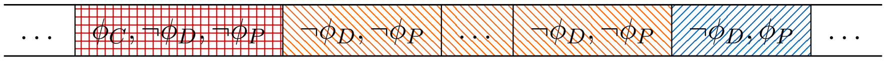
Definition 1 (Conditional Prohibition)**.** *A* conditional prohibition *(over L) is a tuple (ϕ_C, ϕ_P, ϕ_D), where ϕ_C, ϕ_P and ϕ_D are propositional formulas, expressed in Disjunctive Normal Form (DNF), over L.*

We refer to ϕ_C as the (detachment) *condition* of the norm, and ϕ_D as the *deadline*. ϕ_P denotes a *target state* that is prohibited to occur after a state where the condition of the norm ϕ_C holds, and before a state where the deadline ϕ_D holds (the norm "expires").

Example. The norm *"if a car enters the 2nd km of the highway, it is prohibited from driving faster than 70 km/h until it reaches the 7th km of the highway"* can be represented as a conditional prohibition $(km_2 \wedge car, sp_{70}, km_7)$. The components $\phi_C = km_2 \wedge\ car$, $\phi_P = sp_{70}$ and $\phi_D = km_7$ are propositional formulas from the propositional language L with propositional variables $V = VP \cup VS \cup VT$ defined above and expressed in DNF.

Definition 2 (Violation of a Prohibition)**.** *A conditional prohibition (ϕ_C, ϕ_P, ϕ_D) is violated by a trace $(s_1, s_2, \ldots, s_m)$ if there are i, j with $1 \leq i \leq j \leq m$ such that ϕ_C is true at s_i, ϕ_P is true at s_j, and there is no k with $i \leq k \leq j$ such that ϕ_D is true at s_k.*

In other words, a norm is violated by a trace if the states in the trace exhibit a pattern of the following type: a state where the norm is detached (red grid, in the illustration below) is followed by a number of states (possibly none) where neither the prohibition is violated nor the deadline is reached (north west orange lines), after which there is a state where the deadline is still not reached, but the prohibition is violated (north east blue lines). Note that the state where the prohibition is violated may be the same state where the norm is detached (not illustrated below, which considers the case where the three types of states are distinct). Also note that the violation of a norm does not distinguish between a single or multiple violations, i.e., a trace violates a prohibition if at least one violation occurs.



[2] Based on the relative cost of False Positive and False Negative, other metrics could be used to assess the alignment of a norm instead of accuracy (e.g., F-measure). *DDNR* supports this and, in different contexts, *DDNR* can be tuned with the most suitable metric.

Violation conditions of conditional norms can be expressed in Linear Time Temporal Logic (LTL) and evaluated on finite traces in linear time (Alechina et al., 2014). Such combination of simplicity of evaluation and expressiveness, which allows to describe temporal patterns of behaviors (frequent, for example, in traffic domains), motivates us to use conditional norms in this paper. Moreover, conditional prohibitions have an intuitive meaning, which enables us to propose intuitive operations of revision of their components.

3. *DDNR*: Data-Driven Norm Revision

In this section, we introduce *DDNR*, our proposed Data-Driven Norm Revision approach. Given a set of norms N and a data set Γ of traces labeled w.r.t. the MAS objectives, *DDNR* synthesises revised norms that are better aligned with the MAS objectives with respect to Γ. *DDNR* consists of two steps.

- *Synthesis step.* *DDNR* synthesises a set of candidate revisions of the norms in N. We discuss the synthesis step in Sections 3.1 and 3.2, where first we describe two data-driven procedures of revision of DNF propositional formulas, and then we show how to combine and use such operations to generate candidate revised norms.
- *Selection step.* *DDNR* selects a set of revised norms from the candidate norms obtained in the *synthesis step*. We describe the selection step in Section 3.3, where we characterize the concept of *alignment* of a norm with the MAS objectives by means of statistical metrics (in particular, *accuracy*) that can be calculated on the data set of traces Γ. The selected norms will be those that are most aligned with the MAS objectives, i.e., those with highest accuracy.

In the following, we first consider, for simplicity, the case of one norm n. In Section 3.4, we show how the approach applies to multiple norms and present the *DDNR* algorithm.

3.1 Data-Driven Procedures to Make a Formula More or Less Specific

We introduce two data-driven procedures, called MoreSpec and LessSpec, that can be used to make a propositional formula ϕ, expressed in DNF, respectively more and less specific. A formula ϕ is more specific than a formula ψ w.r.t. a data set of traces Γ if and only if the set of states from traces in Γ in which ϕ holds is a subset of the states in which ψ holds. Similarly, ϕ is less specific than ψ iff the set of states from traces in Γ where ϕ holds is a superset of the states where ψ holds.

The inputs of these procedures are (i) a formula ϕ, expressed in DNF, that needs to be revised into a less or more specific one (for example, this formula could be the condition ϕ_C of the norm being revised); (ii) a set of states S, belonging to traces from a data set Γ, from which propositions can be extracted to use for synthesising a new formula; and (iii) a set V of all possible propositions that can be used for ϕ and its revisions (note that the condition of a norm can be expressed using propositional variables that are different from that of the deadline).

Algorithm 1 (MoreSpec) constructs *more specific* formulas than the input formula ϕ, i.e., formulas that are true in *a subset* of the states where ϕ is true. First (line 3), it retrieves *rel_prop*, the set of propositions from V that occur in at least one of the states in S. $prop(S, V) = \{p \mid p \in V, \exists s \in S : p \in s\}$ indicates a function that returns the set of propositions from V that occur in at least one of the states in $S \subseteq \mathcal{S}$. Algorithm 1 then calls, in line 4, a function $\textsc{buildConj}$, which constructs a set C of conjunctions of propositions from rel_prop (e.g., given two propositions p and q, it constructs $C = \{p, q, p \wedge q\}$).

Since the maximum number of possible conjunctions of propositions from rel_prop is $2^{|rel\_prop|}$, i.e., it is exponential in the number of propositions in rel_prop, function $\textsc{buildConj}$ constrains the number of synthesized conjunctions to a constant to limit the exponential growth of set C. This can be done in a general way (e.g., by immediately returning after the first x conjunctions that are created, or by uniformly sampling a limited number of propositions from rel_prop to use to create conjunctions) or in an informed way by using domain knowledge to consider only some of the propositions in rel_prop (e.g., leveraging knowledge about the semantics of the propositions). Another analogous general strategy is to bound the maximum number of conjuncts in each conjunction, similarly to standard metrics of minimality in ILP (Corapi et al., 2011). In our empirical evaluation, we both constrain the maximum number of conjunctions and use some domain knowledge to filter conjunctions in an informed way (more details in Section 4.1).

Algorithm 1 MoreSpec

```
1:  Input: formula $\phi$ in DNF; set of states $S$ from $\Gamma$, set of propositional variables $V$
2:  Output: set of formulas $msf$ more specific than $\phi$
3:  $rel\_prop \leftarrow prop(S, V)$
4:  $C \leftarrow \{\top\} \cup \textsc{buildConj}(rel\_prop)$
5:  $msf \leftarrow \emptyset$
6:  for $c \in C$ do
7:      $f \leftarrow \bot$
8:      for $d \in \textsc{disjuncts}(\phi)$ do
9:          $f \leftarrow f \vee (d \wedge c)$
10:     $msf \leftarrow msf \cup \{f\}$
11: return $msf$
```

We note that, clearly, limiting the size and characteristics of set C as described above comes at the expenses of completeness w.r.t. the set of all possible revised formulas that could be synthesised. This could later result in lower accuracy of the revised norms w.r.t. to the data (the worst case is that no norm that is more accurate than the original one is found). However, doing so allows to provide a tractable solution to the problem being considered. We note, furthermore, that, even in the case of exhaustive search, no guarantee is given that a more accurate norm is found given a data set of traces. A more in depth discussion of this topic is provided in Section 5.

After obtaining the set C, Algorithm 1 constructs a set of new formulas msf (lines 5-10)[3] *more specific* than ϕ by adding each conjunction $c \in C$ in conjunction with each disjunct of the original ϕ, creating a new disjunction f. The resulting set of disjunctions msf is a set of formulas that are more specific than ϕ (incl. ϕ itself). Note that this follows from the

[3] $\bot$ and $\top$ are the atomic propositions indicating falsehood and truth, respectively.

fact that every new constructed conjunction will be composed of at least the propositions contained in ϕ (i.e., it will be true in a state s only if ϕ is already true in s).

Example. Let V be the propositional variables defined in Section 2.1, $\phi = km_2$, and $S = \{s\}$, with $prop(\{s\}, V) = \{km_1, km_2, sp_{10}, \ldots, sp_{22}, car\}$. Examples of formulas more specific than ϕ that can be obtained via moreSpec(ϕ, S, V), i.e., that are in the set msf returned by the function, include $(km_2), (km_2 \wedge sp_{22}), (km_2 \wedge car), (km_2 \wedge sp_{22} \wedge car), (km_2 \wedge sp_{21} \wedge car), (km_2 \wedge sp_{15} \wedge sp_{18}), (km_2 \wedge sp_{15}), (km_2 \wedge sp_{18}), (km_2 \wedge sp_{15} \wedge car)$, etc.[4]

Algorithm 2 LessSpec

```
1: Input: formula $\phi$ in DNF; set of states $S$ from $\Gamma$, set of propositional variables $V$
2: Output: set of formulas $lsf$ less specific than $\phi$
3: $rel\_prop \leftarrow prop(S, V)$
4: $C \leftarrow \{\bot\} \cup$ buildConj$(rel\_prop)$
5: $lsf \leftarrow \{(\phi \vee c) \mid c \in C\}$
6: return $lsf$
```

Algorithm 2 (LessSpec) constructs *less specific* formulas than the formula ϕ given as input, i.e., formulas that are true in *a superset* of the states where ϕ is true. The algorithm is slightly different from Algorithm 1. Here, we create a formula (a conjunction) c which will be true in some of the states in S, and we synthesise a new formula $\phi \vee c$ (less specific than ϕ) where c is *in disjunction* with ϕ. We do not need to unroll the original formula ϕ as done in the loop in line 8 of Algorithm 1, since here we introduce another disjunct to ϕ, which is already in DNF. The resulting set of less specific formulas lsf is constructed as per line 5. Note that, by definition of state, if ϕ is true in a set of states S_ϕ, and c is true in a set of states S_c, a formula $f = \phi \vee c$ will be true in the (larger or equal) set of states $S_\phi \cup S_c$, thereby making f less specific than ϕ.

Example. Let V, $\phi = km_2$, and S be the same as per the example given for Algorithm 1. Examples of formulas less specific than ϕ that can be obtained by invoking lessSpec(ϕ, S, V) include $(km_2), (km_2 \vee sp_{22}), (km_2 \vee car), (km_2 \vee (sp_{22} \wedge car)), (km_2 \vee (sp_{21} \wedge car)), (km_2 \vee (sp_{15} \wedge sp_{18}))$.

3.2 Revising the Norm: the *Synthesis Step*

We illustrate the first step of *DDNR*, the *synthesis step*, which is described by Algorithm 3 (Synthesis). Consider a norm $n = (\phi_C, \phi_P, \phi_D)$ to be revised. We distinguish three types of revisions of a norm: *alteration*, *weakening*, and *strengthening*. In general, altering a norm n changes the set of behaviors prohibited by n in an arbitrary way, while weakening and strengthening a norm n create new norms that prohibit, respectively, a subset and a superset of the behaviors prohibited by n. In particular, we have:

Alteration – Prohibiting different behaviors. An alteration of a norm n is a new norm n' that prohibits a *different* set of behaviors than n. An alteration of n can be realized by making at least one of the components of n, that is the condition, the target state or the deadline, either more or less specific. For example, an alteration of n is a new norm

[4] Note that, as per Section 2.1, the semantics of $sp_x \wedge sp_y$ is "the speed is both higher than x and y km/h".

$n' = (\phi'_C, \phi'_P, \phi'_D)$ such that ϕ'_C is less specific than ϕ_C, while ϕ'_P and ϕ'_D are more specific than ϕ_P and ϕ_D, respectively.

Weakening – Prohibiting fewer behaviors. A weakening of a norm n is a special case of alteration of n: the new norm n' prohibits a *subset* of the behaviors prohibited by n. To weaken a norm, it is necessary to do at least one of the following operations, each causing n' to prohibit fewer behaviors: (i) making the *condition more specific*, so that the norm is detached in fewer states; (ii) making the *target state more specific*, so that fewer target states are prohibited; or (iii) making the *deadline less specific*, so that the norm "expires" in more states.

Strengthening – Prohibiting more behaviors. A strengthening of a norm n is another special case of alteration of n, leading to a n' that prohibits a *superset* of the behaviors prohibited by n. To strengthen a norm, it is necessary to do at least one of the following operations, each causing n' to prohibit more behaviors: (i) making the *condition less specific*, so that the norm is detached in more states; (ii) making the *target state less specific*, so that more target states are prohibited; or (iii) making the *deadline more specific*, so that the norm "expires" in fewer states.

Algorithm 3 (Synthesis) takes as input a set of traces Γ, a norm $n = (\phi_C, \phi_P, \phi_D)$ to revise, the *type* of norms to synthesise (alteration, weakening or strengthening), and the propositional variables related to the different components of n. As output, Synthesis returns a triple $\langle cand\_c, cand\_p, cand\_d \rangle$ composed of three sets of formulas, which are the sets of candidate revisions for the components ϕ_C, ϕ_P and ϕ_D of the original norm n determined by making the norm components more or less specific, by invoking the procedures described in Section 3.1. In principle, by creating the set of all possible combinations of formulas in *cand_c*, *cand_p* and *cand_d*, it is possible to obtain a set of norms which are revisions of the original norm n. We call such a set $\mathcal{R}(n)$. Moreover, when referring to a particular type of revision, we will use $\mathcal{A}(n)$, $\mathcal{W}(n)$ and $\mathcal{S}(n)$, for alteration, weakening and strengthening respectively, instead of $\mathcal{R}(n)$. We highlight, however, that the *DDNR* algorithm (described in Algorithm 5) does not require to create such a set of all possible combinations.

The three types of revision are visible in Algorithm 3 in lines 6, 10, and 14. In each case (e.g., lines 11–13 for the case of *weakening*), the algorithm synthesises a number of candidate new components of the norms by using Algorithms 1 and 2 described in the previous section. In the case of weakening, for example, the set of candidate new conditions (*cand_c*) is composed of the more specific conditions obtained with MoreSpec. Algorithms 1 and 2 are invoked with different input parameters based on the type of component (e.g., the condition) and on the type of new formulas we wish to obtain (e.g., *more specific* conditions). For instance, in line 11, MoreSpec is invoked with the current condition ϕ_C, a set of "Condition" States CS which we describe below, and the set V_C of the propositional variables that are relevant for the condition of the norm. The idea is to modify ϕ_C by making it more specific to a set of states CS carefully selected from traces in Γ so that the resulting norm will be *detached* in fewer states, thus prohibiting fewer traces (i.e. weakening the norm). In line 19, finally, the three sets of revised components are returned. These sets, if combined together, represent the final set $\mathcal{R}(n)$ of candidate revisions of the input norm n.

Algorithm 3 Synthesis

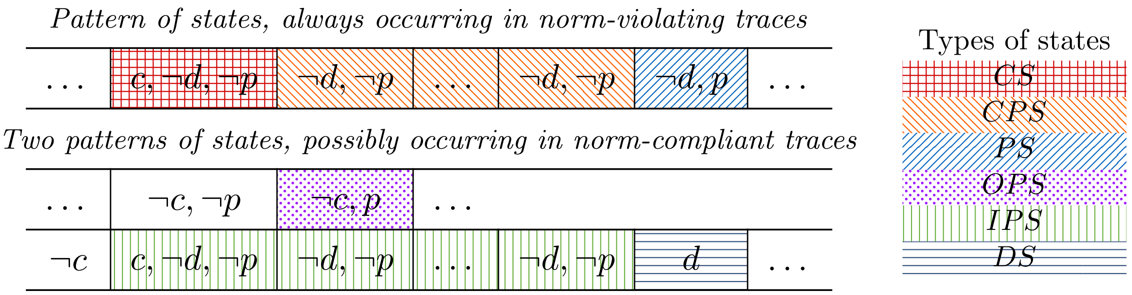
```
1:  Input: data set $\Gamma$; norm $n = (\phi_C, \phi_P, \phi_D)$; type of revision type (alteration, weakening,
    or strengthening); $V_C$, $V_P$, $V_D$ propositional variables for condition, target state and
    deadline
2:  Output: triple of sets of candidate revisions of the components of norm $n$
3:
4:  $CS, OPS, PS, IPS, CPS, DS \leftarrow \textsc{getStates}(\Gamma, n)$
5:
6:  if type is alteration then
7:      $cand\_c \leftarrow \textsc{MoreSpec}(\phi_C, CS, V_C) \cup \textsc{LessSpec}(\phi_C, OPS, V_C)$
8:      $cand\_p \leftarrow \textsc{MoreSpec}(\phi_P, PS, V_P) \cup \textsc{LessSpec}(\phi_P, IPS, V_P)$
9:      $cand\_d \leftarrow \textsc{MoreSpec}(\phi_D, DS, V_D) \cup \textsc{LessSpec}(\phi_D, CPS, V_D)$
10: if type is weakening then
11:     $cand\_c \leftarrow \textsc{MoreSpec}(\phi_C, CS, V_C)$
12:     $cand\_p \leftarrow \textsc{MoreSpec}(\phi_P, PS, V_P)$
13:     $cand\_d \leftarrow \textsc{LessSpec}(\phi_D, CPS, V_D)$
14: if type is strengthening then
15:     $cand\_c \leftarrow \textsc{LessSpec}(\phi_C, OPS, V_C)$
16:     $cand\_p \leftarrow \textsc{LessSpec}(\phi_P, IPS, V_P)$
17:     $cand\_d \leftarrow \textsc{MoreSpec}(\phi_D, DS, V_D)$
18:
19: return $\langle cand\_c, cand\_p, cand\_d \rangle$
```

We distinguish six sets of states (the states *CS*, *OPS*, etc. in line 4) that can be obtained from the traces in data set Γ w.r.t. the norm n. Three of these sets, namely CS, PS and CPS, are defined with respect to norm-violating traces and they are used to weaken n. The remaining sets, i.e., OPS, IPS and DS, are defined with respect to norm-compliant traces and they are used to strengthen n. All the six sets are used to alter n. Below, we formally define and explain the details of each of these six sets.

Algorithm 4 provides a formal definition of all the states. The six states are defined by exploiting three patterns of states (illustrated in Figure 3a) that are, or can be, exhibited by traces in Γ due to the semantics of the violation of a norm (e.g., the pattern of states that is always exhibited in prohibited traces as discussed in Section 2.1). In Figure 3, furthermore, we report an example of data set Γ whose traces are labeled according to a norm n and to 7 examples of revisions[5] of n obtained with Algorithm 3.

CS **(*Condition States*, red grid in Figure 3a)** is the set of states in norm-violating traces where n is detached and, after that, a prohibited target state is encountered before reaching the deadline. At least one such state exists in each norm-violating trace, as per Definition 2. Synthesis extracts propositions from the states in CS to construct new *conditions* that are more specific than the current one (e.g., lines 7 and 11), so that the resulting norms will be detached in, thus prohibit, fewer traces currently norm-violating.

[5] Note in Figure 3 that, as discussed above, weakenings of n prohibit no more traces than n, strengthenings prohibit no fewer traces than n, and alterations of n prohibit a different set of traces than n.



(a) Three possible patterns of states that can appear in traces in Γ. Different states are coloured based on their membership to a type of states (indicated by their color, as per the legend on the right) used in Algorithm 3, and labeled w.r.t. the truth value of the components of a generic norm $n = (\phi_c, \phi_p, \phi_d)$ (for readability we write, e.g., c instead of ϕ_c). The first pattern, described in Section 2.1, always appears in norm-violating traces. The second pattern can appear in any kind of norm-compliant trace, regardless of whether the norm is detached, and describes a case where a target state (purple dots) is encountered before the norm is detached (possibly after the deadline was reached). The third pattern can appear in norm-compliant traces where the condition is detached and, after that, the deadline is reached (grey horizontal lines) after encountering a number of states that are not prohibited (green vertical lines).

Data set Γ				Norm n	Examples of weakening of n: $n_1 = (a \wedge b, c, e)$ $n_2 = (a, c \wedge d, e)$ $n_3 = (a, c, e \vee a)$			Examples of strengthening of n: $n_4 = (a \vee c, c, e)$ $n_5 = (a, c \vee i, e)$ $n_6 = (a, c, e \wedge f)$			Examples of alteration of n: $n_7 = (a \vee c,$ $c \vee j,$ $e \vee b)$
Trace	s_1	s_2	s_3	(a, c, e)	n_1	n_2	n_3	n_4	n_5	n_6	n_7
γ_1	a,b	c,d	e,f	✓	✓	✓		✓	✓	✓	
γ_2	a	c,d	k,l	✓		✓		✓	✓	✓	✓
γ_3	a,b	i,j	e,f						✓		
γ_4	a,b	i.j	k,l						✓		
γ_5	g,h	c,d	e,f					✓			✓
γ_6	g,h	c,d	k,l					✓			✓
γ_7	g,h	i,j	e,f								
γ_8	g,h	i.j	k,l								

(b) Data set Γ composed of 8 finite traces, each made of 3 states (columns s_1-s_3). States are colored based on their types, as per Figure 3a, w.r.t. norm n, and, on the right, traces are labeled according to a norm n and to 7 examples of revisions of n obtained from Algorithm 3: the cells containing ✓ indicate that the corresponding trace is norm-violating, empty cells indicate that the trace is norm-compliant.

Figure 3: (a) Three possible patterns of states that can appear in traces in a data set Γ; and (b) an example of Γ where traces' states are colored according to their type as per sub-figure 3a, and traces are labeled according to different norms.

CPS (***In-Between Condition and Prohibited States*, orange north west lines in Figure 3a)** is the set of states that are in between the state where the condition holds and the state where the prohibition holds in norm-violating traces. Synthesis extracts propositions from the states in *CPS* to construct less specific *deadlines* than the current one, so that the resulting norms will expire also in some state in *CPS* (which are in between the condition and the target state holds), thus classifying more traces as norm-compliant.

PS (***Prohibited States*, blue north east lines in Figure 3a)** is the set of states in norm-violating traces where ϕ_P holds after the norm is detached and before the deadline is reached. Synthesis extracts propositions from the states in *PS* to construct more specific prohibited *target states*, so that fewer states currently prohibited will violate the resulting norms, thus making norm-compliant more traces that are currently norm-violating.

OPS (***Outer Prohibited States*, purple dots in Figure 3a)** is the set of states in norm-compliant traces where we know that ϕ_P holds, if any. If such a state exists, then it is *not* in between a state where the condition holds and another where the deadline holds, otherwise the trace would violate n. Synthesis extracts propositions from the states in *OPS* to construct less specific *conditions* than the current one, so that the resulting norms will apply also to states in *OPS*, where, if ϕ_P is not changed, the norm is violated, thus prohibiting more traces that are currently norm-compliant.

Algorithm 4 getStates

```
Input: data set $\Gamma$; norm $n = (\phi_C, \phi_P, \phi_D)$;
Output: six sets of states from traces in $\Gamma$
$CS \leftarrow \{ s_i \mid \exists(s_1, \dots, s_i, \dots s_m) \in \Gamma \;\; s.t. \;\; s_i \vDash \phi_C,$
        $\exists j \mid i \leq j \leq m \; s.t. \; s_j \vDash \phi_P,$
        $\nexists k \mid i \leq k \leq j \; s.t. \; s_k \vDash \phi_D \}$
$CPS \leftarrow \{ \{s_i \mid \exists(s_1, \dots, s_i, \dots s_m) \in \Gamma \;\; s.t. \;\; \exists j, k \mid 1 \leq j < i < k \leq m \; s.t.$
        $s_j \vDash \phi_C, s_k \vDash \phi_P,$
        $\nexists l \mid j \leq l \leq k \; s.t. \; s_l \vDash \phi_D \}$
$PS \leftarrow \{ s_i \mid \exists(s_1, \dots, s_i, \dots s_m) \in \Gamma \;\; s.t. \;\; s_i \vDash \phi_P,$
        $\exists j \mid 1 \leq j \leq i \; s.t. \; s_j \vDash \phi_C,$
        $\nexists k \mid j \leq k \leq i \; s.t. \; s_k \vDash \phi_D \}$
$OPS \leftarrow \{ s_i \mid \exists\gamma = (s_1, \dots, s_i, \dots s_m) \in \Gamma \;\; s.t. \;\; s_i \vDash \phi_P,$
        $n$ is not violated on $\gamma \}$
$IPS \leftarrow \{ s_i \mid \exists\gamma = (s_1, \dots, s_i, \dots s_m) \in \Gamma \;\; s.t. \;\; n$ is not violated on $\gamma,$
        $\exists j, k \mid 1 \leq j \leq i < k \leq m \; s.t.$
        $s_j \vDash \phi_C,$
        $s_k \vDash \phi_D \}$
$DS \leftarrow \{ s_i \mid \exists\gamma = (s_1, \dots, s_i, \dots s_m) \in \Gamma \;\; s.t. \;\; n$ is not violated on $\gamma,$
        $s_i \vDash \phi_D$
        $\exists j \mid 1 \leq j \leq i \; s.t. \; s_j \vDash \phi_C,$
        $\nexists k \mid j \leq k < i \; s.t. \; s_k \vDash \phi_D \}$
return $CS, OPS, PS, IPS, CPS, DS$
```

IPS (***I**nner **P**ermitted **S**tates***, green vertical lines in Figure 3a**) is the set of states that are in between states where the condition and the deadline hold in norm-compliant traces, if any. Synthesis extracts propositions from the states in *IPS* to construct less specific prohibited *target states* than the current one, so that the resulting norms will also prohibit states currently in norm-compliant traces.

DS (***D**eadline **S**tates***, grey horizontal lines in Figure 3a**) is the set of states where ϕ_D holds (the deadline is reached), if any, in norm-compliant traces after the norm is detached. Synthesis extracts propositions from the states in *DS* to construct more specific *deadlines* than the current one, so that the resulting norms will expire in fewer states, potentially prohibiting more traces currently norm-compliant.

3.3 Choosing the New Norm: the *Selection Step*

The *synthesis step* of *DDNR* produces a triple $\langle cand\_c, cand\_p, cand\_d \rangle$ representing the set $\mathcal{R}(n)$ of candidate revisions a norm n. In this section, we discuss the *selection step*, which chooses the new norm from $\mathcal{R}(n)$.

We recall that, as discussed in Section 2.1, by analysing a confusion matrix that characterizes the traces in a data set w.r.t. the classification provided by a norm, we can determine the number of classification errors of a norm, the type of errors (i.e., whether negative traces are considered compliant more often than positive traces are considered violating), and we can determine whether a norm is better (aligned with the MAS objectives) than another.

We characterize the concept of *alignment* of a norm with the MAS objectives using the *accuracy* metric, defined as $acc(n, \Gamma) = \frac{TP+TN}{|\Gamma|}$, with TP and TN being the number of true positives and true negatives obtained with a norm n on the data set of traces Γ.

Therefore, the *selection step* aims at choosing the combination of revised components from $cand\_c$, $cand\_p$ and $cand\_d$ (i.e., the revision of n) with highest accuracy. In other words, given a set of revised norms $\mathcal{R}(n)$, we choose, as a revision of n, the norm with highest accuracy, i.e., $n^* = argmax_{n' \in \mathcal{R}(n)}\ acc(n', \Gamma)$.

As an example, consider the three confusion matrices in Table 2, which are determined by the three norms in $\mathcal{R}(n) = \{n_1, n_2, n_3\}$ obtained with Algorithm Synthesis on a data set Γ that consists of 12 traces, 6 of which are positive traces and 6 are negative. Norm n_1 classifies correctly 75% of the traces making few errors among the positive traces and slightly more errors among the negative traces. Norm n_2 classifies correctly only 50% of traces. n_2 is weaker than n_1 and, thus, classifies many negative traces as norm-compliant. Finally, n_3 classifies correctly 66% of traces. This norm is stricter than the others, and it captures all negative traces. This, however, comes at the cost of misclassifying many positive traces. In the *selection step*, we can choose n_1 as the best revised norm in $\mathcal{R}(n)$, as its accuracy is the highest. Again, note that based on the particular problem being considered, and on the relative cost of False Positive and False Negative, other metrics could be used as an alternative (or in addition) to accuracy. The metric can be considered a parameter of *DDNR*. In this work, we choose accuracy as a metric since we focus on the introduction of novel algorithms, we do not make specific claims about the cost of FP and FN, and we consider data sets where the traces are evenly split among positive and negative.

Table 2: Accuracy of three examples of norms.

	n_1	n_2	n_3
$\begin{bmatrix} \mathrm{TP} & \mathrm{FN} \\ \mathrm{FP} & \mathrm{TN} \end{bmatrix}$	$\begin{bmatrix} 5 & 1 \\ 2 & 4 \end{bmatrix}$	$\begin{bmatrix} 4 & 2 \\ 4 & 2 \end{bmatrix}$	$\begin{bmatrix} 2 & 4 \\ 0 & 6 \end{bmatrix}$
acc	0.75	0.5	0.66

3.4 Multiple Norms

Revising a set of norms $\mathcal{N}$ corresponds to generating a set of norms $\mathcal{N}'$ such that at least one of the norms in $\mathcal{N}$ is revised. A set of norms $\mathcal{N}$ is weakened (strengthened) when only weakening (strengthening) is applied, otherwise the set is altered. In Section 2.1, we interpreted a norm as a binary classifier that distinguishes norm-compliant from norm-violating traces. We can generalize this concept to a set of norms. We interpret the set of norms as a multi-label binary classifier, where multiple binary labels can be assigned to each trace. Each norm assigns a different binary label to a trace.

Consider, for example, a set $\mathcal{N} = \{n_1, n_2\}$ composed of two norms and a data set of traces Γ. A trace $\gamma \in \Gamma$ can be compliant with n_1 and it can violate n_2, thereby two labels n_1 *compliant* and n_2 *violating* are assigned to the trace γ. Unlike the case of a single norm, where a trace could be either correctly or wrongly classified, a trace can now be also partly correctly classified. This happens, for example, if a trace is a positive trace and it complies with one norm but it violates another norm. This situation is illustrated in Figure 4. Suppose we aim to revise $\mathcal{N}$ by strengthening n_1 and weakening n_2. One straightforward way to apply the proposed approach is to order the norms in some way and revise them one by one, performing the *synthesis* and *selection steps* for each norm independently.

$$\begin{array}{ll|cccc|}
 & & \multicolumn{2}{c}{n_1\ \textit{compliant}} & \multicolumn{2}{c}{n_1\ \textit{violating}} \\
 & & n_2\ \textit{compliant} & n_2\ \textit{violating} & n_2\ \textit{compliant} & n_2\ \textit{violating} \\
\text{objectives} & \text{positive} & \mathrm{PFC} & \mathrm{PPC}_1 & \mathrm{PPC}_2 & \mathrm{PFW} \\
 & \text{negative} & \mathrm{NFW} & \mathrm{NPC}_2 & \mathrm{NPC}_1 & \mathrm{NFC}
\end{array}$$

Figure 4: A confusion matrix for the case of two norms n_1 and n_2. *PFC* stands for *Positive Fully Correct* (i.e., number of positive traces in a data set Γ correctly classified by both norms); PPC_i stands for *Positive Partly Correct*, with i indicating the id of the norm that classifies correctly; *PFW* for *Positive Fully Wrong*. *NFW*, NPC_i and *NFC* are analogous for the Negative traces.

Revising each norm independently may lead to a set of new norms such that, while each norm is better aligned with the MAS objectives, their combination diminishes the number of fully correct classifications compared to the original set. Figure 5 shows an example (artificial, for the sake of illustration) where a set of norms $\mathcal{N} = \{n_1, n_2\}$ is revised by weakening n_1 and strengthening n_2, obtaining a new set $\mathcal{N}'$. While the number of correctly classified traces by each norm increases after the revision, the number of traces correctly classified by both of them at the same time decreases, increasing instead the number of partly correct classifications. Comparing the values inside and outside the brackets in Figure 5b, we notice that after the revision all traces are only partly correctly classified.

Data set Γ		$\mathcal{N}$		$\mathcal{N}'$	
Trace	Label	n_1	n_2	n_1	n_2
γ_1	positive	violating	violating	compliant	violating
γ_2	negative	compliant	compliant	compliant	violating
γ_3	positive	violating	compliant	compliant	violating
γ_4	negative	violating	violating	compliant	violating

(a)

$$\begin{array}{cc|cccc|} & & \multicolumn{2}{c}{n_1\ \textit{compliant}} & \multicolumn{2}{c}{n_1\ \textit{violating}} \\ & & n_2\ \textit{compliant} & n_2\ \textit{violating} & n_2\ \textit{compliant} & n_2\ \textit{violating} \\ \text{objectives} & \textit{positive} & 0(0) & 0(2) & 1(0) & 1(0) \\ & \textit{negative} & 1(0) & 0(2) & 0(0) & 1(0) \end{array}$$

(b)

Figure 5: Example of classification of four traces by a set of norms $\mathcal{N}$ and by its revision $\mathcal{N}'$ (a) and the corresponding confusion matrix (b). The values in the matrix in between brackets refer to the set $\mathcal{N}'$.

If having fully correctly classified traces is not important, e.g., if obeying one or some of the norms is sufficient for achieving the MAS objectives, then revising each of them individually may be a good strategy and the alignment of the norm set can be determined as the average alignment of all the norms. If, conversely, it is important to have fully correctly classified traces, instead of revising each norm independently, we can search for a combination of norms that minimizes the combined errors. Similarly to the case of one norm, we can look for the *set* of norms that is most aligned with the MAS objectives. This time, however, the alignment of the norm set must be determined w.r.t. the whole set of norms at once. A direct generalization of the accuracy to multi-label problems[6] is defined as the average across all traces of the proportion of the predicted correct labels to the total number (predicted and actual) of labels for each trace. Equation 2 reports its formalization. Z_i is the vector of labels predicted by the norms in $\mathcal{N}$ for a trace i (e.g., a label n_1 *compliant* expresses that a trace is compliant with norm n_1), and Y_i is the vector of correct labels for trace i according to the MAS objectives (e.g., if the trace is *negative* w.r.t. the MAS objectives, then the correct label is *norm-violating*). Note that, exclusively in Equation 2, we follow the notation from (Sorower, 2010) where the intersection and union symbols (i.e.,

[6] Other metrics for assessing the output of a multi-label classifier include the Hamming Loss (Schapire and Singer, 2000) or the Weighted Kappa (Cohen, 1968). As per the case of a single norm, all these metrics (and others) can be used in *DDNR* to assess the alignment of a norm set with the MAS objectives.

$\cap$ and $\cup$) indicate respectively the AND and OR operation between every pair of labels in the two vectors of labels Y_i and Z_i. Each operation results in a boolean vector and $|\cdot|$ indicates the number of elements with value *true* in such a vector.

$$\mathit{ml\text{-}acc} = \frac{1}{|\Gamma|} \sum_{i=1}^{|\Gamma|} \frac{|Y_i \cap Z_i|}{|Y_i \cup Z_i|} \tag{2}$$

Therefore, given a set of norms $\mathcal{N}$, we can order the norms in some way, perform for each of them the *synthesis step*, obtaining a list of possible revisions

$$\mathcal{R}(\mathcal{N}) = \left[\langle \mathit{cand\_c}, \mathit{cand\_p}, \mathit{cand\_d}\rangle_{n_1}, \ldots, \langle \mathit{cand\_c}, \mathit{cand\_p}, \mathit{cand\_d}\rangle_{n_{|\mathcal{N}|}}\right]$$

and perform the *selection step* to choose the combination of norms that maximizes metric *ml-acc*.

When the number of norms grows, an exhaustive search through all possible combinations is infeasible. We can instead adopt a Monte Carlo algorithm and search the space of all possible combinations of revised norms by randomly sampling k possible combinations[7].

Algorithm 5 summarizes *DDNR*, the data-driven norm revision mechanism proposed in this paper. *DDNR* takes as input a data set of traces Γ labelled by the MAS objectives evaluator, a set of norms $\mathcal{N}$ to be revised, the type of revision required for each norm, the propositional variables to use for revising the components of each norm, and the required number of samples k for the Monte Carlo search. As output, *DDNR* returns a list of revised norms, one for each norm in $\mathcal{N}$.

Function SYNTHESIS (line 6) performs the *synthesis step* for a norm n as per Section 3.2, returning a triple $\langle \mathit{cand\_c}, \mathit{cand\_p}, \mathit{cand\_d}\rangle$ of candidate revisions of the components of the original norm n. This function is applied for all norms in $\mathcal{N}$ (line 5) so to populate the list $\mathcal{R}(\mathcal{N})$ of triples, one for each norm. Function RANDOMSAMPLE (line 7) takes as input such list $\mathcal{R}(\mathcal{N})$ and the number k of required samples, and randomly samples each of the sets in each triple in the list, so to obtain a set of k lists, each composed of one candidate revision for each norm in $\mathcal{N}$. Random sampling can be done in constant time via standard procedures[8], and can be performed *without replacement* so to return a set of unique items.

The algorithm returns the best candidate revision found in the k samples (line 8). In the reported algorithm, the best candidate is that with highest ML-ACC (the multi-label accuracy). We assume that all norms are equally important, but we note that extending the algorithm to consider the importance of each norm (omitted here) is straightforward.

[7] Note that a sampling strategy is justified by the fact that, as we discuss in Section 5, even an exhaustive search among the set of all combinations of synthesised norms would not guarantee to find a perfect solution (i.e., a set of norms that is perfectly aligned with the MAS objectives), because Algorithm SYNTHESIS adopts an heuristic approach that relies, for generating revised norms, (i) on the available data and (ii) on the definition of the function BUILDCONJ, which might discard some better conjunctions of propositions to keep the algorithm tractable.

[8] See for example the Python `random.sample()` function https://docs.python.org/3/library/random.html.

Algorithm 5 *DDNR*

```
Input: data set $\Gamma$; set of norms $\mathcal{N}$; list $T$ of $|\mathcal{N}|$ types of revisions, one for each norm in $\mathcal{N}$; list $V$ of $|\mathcal{N}|$ triples $\langle V_n^C, V_n^P, V_n^D \rangle$ of propositional variables, one for each norm $n$ in $\mathcal{N}$; number of samples $k$
Output: ordered list of revised norms selected
                                                        ▷ Synthesis step
$\mathcal{R}(\mathcal{N}) \leftarrow [\ ]$                              ▷ empty list
for all norm $n \in \mathcal{N}$ do
    $\mathcal{R}(\mathcal{N})$.append(synthesis($\Gamma$, $n$, $T_n$, $V_n^C$, $V_n^P$, $V_n^D$))
                                                        ▷ Selection step
candidates ← randomSample($\mathcal{R}(\mathcal{N})$, $k$)
return $argmax_{cand \in candidates \cup \{\mathcal{N}\}}$ ml-acc($cand, \Gamma$)
```

3.5 Complexity of *DDNR*

The complexity of the proposed approach depends on the complexity of (A) the synthesis and (B) selection steps.

(A) Synthesis step. The complexity of the synthesis step depends on the size $|\Gamma|$ of the data set Γ, on the number of norms $|\mathcal{N}|$, on the complexity of functions moreSpec and lessSpec, and on the number of times they are invoked for each norm. The complexity of moreSpec and lessSpec depends on the size of the set C of conjunctions constructed via function buildConj, and on the number of disjuncts in each norm being revised. By bounding the maximum size of conjunctions constructed via buildConj to either a constant or to a polynomial in the number of propositions in the states in Γ (let such polynomial be $f(|\Gamma|)$), as described in Section 3.1, the resulting complexity of buildConj, thus the size of the set C, are in the worst case polynomial in $|\Gamma|$, i.e., $O(f(|\Gamma|))$. This is also the complexity of lessSpec. In the case of moreSpec, each element in C is added to every disjunct in the formula being revised. If the maximum number of disjuncts in a component of any norm in $\mathcal{N}$ is the constant m, then the complexity of moreSpec is $O(mf(|\Gamma|))$.

Computing each of the sets of states in Algorithm synthesis for a norm n requires a time that is $O(\lambda|\Gamma|)$, with λ a constant representing the maximum length of (number of states in) a trace in Γ. This is because extracting a particular set of states from a trace requires a time that is linear in the size of the trace[9] and this needs to be done for every trace in the data set. This operation needs to be executed 6 times per norm (one per type of state), resulting in a complexity $O(6\lambda|\Gamma|)$. In the worst case, for the synthesis step we perform an alteration for each of the $|\mathcal{N}|$ norms. An alteration requires to invoke 3 times function moreSpec and 3 times function lessSpec, resulting in a complexity polynomial in the size of Γ and $\mathcal{N}$ (data set and norms).

[9] Similarly to the monitoring of a conditional norm, extracting a set of states from a trace can be done by reading the states in the trace in order while preserving a boolean flag *det* that indicates whether an instance of the norm has been detached in a previous state, storing the relevant states once encountered, and discarding them if all conditions are not met. Please see our online appendix (Dell'Anna et al., 2022a) for detailed algorithms.

(B) Selection step. The complexity of the selection step is polynomial in Γ, in particular $O(k\lambda|\Gamma|)$, since using a measure such as accuracy to calculate the norm alignment has a complexity that is linear in the size of $\lambda\Gamma$, and such operation is performed k (a constant) times, one per each sample.

Table 3 summarizes the complexity of the different algorithms and steps of *DDNR*.

Algorithm/Step	Complexity
buildConj	$O(f(\lvert\Gamma\rvert))$
lessSpec	$O(f(\lvert\Gamma\rvert))$
moreSpec	$O(mf(\lvert\Gamma\rvert))$
SYNTHESIS (alteration of one norm)	$O(6\lambda\lvert\Gamma\rvert + 3(m+1)f(\lvert\Gamma\rvert))$
SYNTHESIS (weakening of one norm)	$O(6\lambda\lvert\Gamma\rvert + 2mf(\lvert\Gamma\rvert) + f(\lvert\Gamma\rvert))$
SYNTHESIS (strengthening of one norm)	$O(6\lambda\lvert\Gamma\rvert + mf(\lvert\Gamma\rvert) + 2f(\lvert\Gamma\rvert))$
Synthesis step (worst case, i.e., alteration of $\lvert\mathcal{N}\rvert$ norms)	$O((6\lambda\lvert\Gamma\rvert + 3(m+1)f(\lvert\Gamma\rvert))\lvert\mathcal{N}\rvert)$
Selection step	$O(k\lambda\lvert\Gamma\rvert)$
DDNR (worst case, i.e., alteration of $\lvert\mathcal{N}\rvert$ norms)	$O((6\lambda\lvert\Gamma\rvert + 3(m+1)f(\lvert\Gamma\rvert))\lvert\mathcal{N}\rvert)$ + $O(k\lambda\lvert\Gamma\rvert)$

Table 3: Complexity of the different algorithms used by *DDNR*. m, k, and λ are constants, respectively representing the maximum number of disjuncts in a component of the norms in $\mathcal{N}$, the number of samples required for the Monte Carlo sampling, and the maximum number of states in any trace in Γ. $f(|\Gamma|)$ represents a polynomial in the size of data set Γ.

4. Empirical Evaluation

We report on experimental results concerning *DDNR*, presented in Section 3. Our experiments aim to provide an empirical answer to the following research questions:

RQ1. *To what extent do the norms synthesised with algorithm* Synthesis *differ from the original norms with respect to the number of FP, FN, TP and TN traces?*

RQ2. *Is the accuracy of the norms revised with DDNR higher than the accuracy of the original norms?*

RQ3. *How does DDNR generalize to previously unseen traces?*

4.1 Experimental Setting

We make use of a traffic simulation of the highway scenario described in Section 2 in order to generate a data set of traces describing the behavior of agents in a MAS. We implement the scenario with the SUMO traffic simulator (Krajzewicz et al., 2012) and we set up our experiments as follows. The source code of our implementation of *DDNR* and the results and material concerning our experiments are available at (Dell'Anna et al., 2022a).

SUMO vehicles. We consider two types of vehicles for our SUMO traffic simulation. A passenger car (simply *car*, in the following) that can reach a maximum speed of 200 *km/h* with a maximum acceleration of $2.6m/s^2$, and a *truck* that can drive up to 130 *km/h*, with a maximum acceleration of $1.3m/s^2$. The emissions of the car refer to a gasoline-driven passenger car Euro norm 4, while the emissions of the truck refer to an average heavy duty vehicle. For the exact details, we refer the reader to the SUMO documentation on vehicles' types[10]. In SUMO, vehicles placed on a road, drive in the direction of the road, and exhibit a number of autonomous behaviors such as overtaking the vehicle ahead, signaling the overtaking, accelerating or braking, keeping a certain safety distance from the vehicles ahead, etc., based on realistic vehicles' models implemented in SUMO.

Agents. We consider two agent types, one per type of SUMO vehicle as described above. Each agent encapsulates a SUMO vehicle, so that some of the default behaviors of the vehicle can be overridden when necessary (in particular, in our case, based on the enforced norms and on the intention of the agent to violate or not the norms). We use a population of agents randomly and uniformly distributed among *cars* and *trucks*. Both agent types aim at driving through the highway section by maximizing their speed. At each simulation step, every agent determines its desired speed according to its internals (i.e., its maximum speed) and to the currently enforced norms. In particular, 75% of the agents are always compliant with the enforced norms, while the remaining 25% will ignore them and focus on maximizing speed. This can be seen as the effect of enforcing the norms by means of sanctions that can be afforded by 25% of agents. Alternatively, this can also be seen as the agents acting according to their types (e.g., using the BOID terminology (Broersen et al., 2001), 75% of the agents are *social* agents giving priority to norm compliance over their desired maximum speed, and 25% of the agents are *selfish*, preferring their desired maximum speed over the norms). Here, we abstract away from the reasons for compliance for the agents, and we simply uniformly draw from a distribution characterized by the percentages above described the compliant and non-compliant agents. Note that, while compliant agents always obey the norms, agents that ignore the norms do not necessary violate them, as their ability to violate the speed limit depends both on their type (which determines the vehicle's maximum speed) and on the surrounding environment (e.g., traffic jams will force agents to slow down, regardless of their preferences). The default behaviors of the SUMO vehicles (e.g., those mentioned above) which are not regulated by the norms are not monitored but they affect the overall behavior of both the individual agents and the MAS. We consider an open MAS such that, at every simulation step, 2 new agents (vehicles) enter the highway section and go through it.

Norms. We consider two types of norms, regulating (i) the maximum speed of the vehicles, and (ii) the minimum safety distance between the vehicles in the highway section. Maximum speed norms are norms (ϕ_C, ϕ_P, ϕ_D) such that ϕ_C, ϕ_P, ϕ_D are propositional formulas from the languages with variables $L_C = VP \cup VT$, $L_{P_{speed}} = VS \cup VT$ and $L_D = VP$, respectively, with VP, VS and VT defined as per Section 2.1. Minimum safety distance prohibitions, instead, are norms (ϕ_C, ϕ_P, ϕ_D) such that ϕ_C, ϕ_P, ϕ_D are propositional formulas from the languages with variables $L_C = VP \cup VT$, $L_{P_{dist}} = VD \cup VT$ and $L_D = VP$, respectively, with VP and VT defined as per Section 2.1 and $VD = \{dist_x \mid 0 \leq x \leq 15 \ \& \ x \in \mathbb{N}\}$, with propositions $dist_x$ denoting that vehicle's distance from the vehi-

[10] As a reference, the default truck model in the SUMO traffic simulator can reach up to 120 g/s of CO_2. More details can be found here: https://sumo.dlr.de/docs/Vehicle_Type_Parameter_Defaults.html

cles ahead is *lower* than x meters[11]. An example of minimum safety distance prohibition is $n_d = (km_2 \wedge car, dist_{10}, km_7)$, which prohibits *cars* from staying closer than 10 meters from the vehicles ahead from km_2 to km_7 of the highway section. We call *SpeedNorms* and *DistanceNorms* the sets of all possible norms that can be defined w.r.t. their languages.

Traces. Throughout the simulations, we collect execution traces that describe the behavior of each agent. An execution trace of an agent is composed of 10 different states, one per each of the 10 kilometers of the highway section. The i-th state of a trace contains information about: (i) the i-th km of the highway, (ii) the maximum speed the agent reached in the i-th km of the highway, (iii) the type of agent (*car* or *truck*), and (iv) the minimum distance the agent maintained from any vehicles ahead in the i-th km of the highway. Each trace is labeled by the *MAS objectives evaluator* w.r.t. the CO_2 emitted by the vehicle on the highway and the time needed to travel from the beginning to the end of the highway section. A trace is labeled as *positive* if the maximum emission of the vehicle from the beginning to the end of the highway section is below a threshold $t_{co2} = 100$ g/s and the travel time is below a threshold $t_{tt} = 450s$ (the time it takes to drive for 10 *km* at 80 *km/h*), and *negative* otherwise. We emphasise once more that *DDNR* is agnostic of such underlying rules for the labeling and it is exposed only to the given labeled traces. Note that the MAS objectives concern a combination of CO_2 emissions and road throughput and they are not expressed in, nor do they explicitly relate to, the language in which the norms are expressed.

***DDNR* Configuration.** In order to provide an accurate discussion of the proposed approach, we do not sample the possible candidate revised norms in Algorithm *DDNR*, but we exhaustively compare all possible candidates (this is analogous to selecting a big enough number of samples k for *DDNR*). This is made possible by keeping the number of candidate norms created in the *synthesis step* manageable. In particular, in our experiments we adopted the following strategy for implementing function buildConj in Algorithms 1 and 2. Given the set of propositions *rel_prop*, buildConj (a) splits the propositions in types, e.g., it separates propositions belonging to VP from those belonging to VT, (b) orders propositions from VP, VD, and VS, if any, alphabetically, (c) extracts the first x propositions from each group in order, and (d) creates all possible combinations of the x propositions. When buildConj is invoked by MoreSpec in step (c), the propositions are ordered in ascending order, while when the function is invoked by LessSpec, the propositions are ordered in descending order. Note that ordering propositions alphabetically is a syntactic operation which can be applied in any domain. In our case study, however, by ordering alphabetically the propositions, we also introduce an ordering that characterizes the domain knowledge (e.g., lower speeds are earlier in a list of speed-related propositions in ascending order). By selecting the first x propositions of each group in order, therefore, we both bound the maximum number of possible conjunctions and we apply a criteria of *similarity* with the current norm based on the available domain knowledge, i.e., we select at most x propositions which are semantically as “close” as possible to the original norm. For example, if the current norm prohibits from speeding over 40 km/h, when looking for candidate more specific prohibitions given the set of states PS, we limit buildConj to

[11] Note that, as they do for their speed, vehicles autonomously adjust also their distance from the vehicle ahead based on their internals. We do not directly affect their decisions about the specific safety distance to maintain. Instead we regulate the *minimum* safety distance.

return the first x propositions from the ordered set of propositions concerning the speed found in the states in PS (e.g., $sp_{41}, sp_{44}, sp_{50}, \dots$). We set $x = 8$ in our experiments. As a consequence, buildConj can never generate more than $2^8 = 256$ conjunctions from propositions belonging to VP, VD, and VS, and never more than $2^2 = 4$ from propositions belonging to VT (because $|VT| = 2$), for a maximum number of possible conjunctions of $2^{10} = 1,024$.

4.2 To What Extent Do the Norms Synthesised With Algorithm Synthesis Differ From the Original Norms With Respect to the Number of FP, FN, TP and TN Traces? (RQ1)

We study the composition of the sets of revised norms generated during the *synthesis step*, as per Section 3.2 for alteration, weakening, and strengthening.

Method. We execute 100 independent traffic simulations. In the *i-th* simulation, we perform three steps.

1. We run the simulation until 1,500 vehicles drive through the highway section under the enforcement of one norm n_i randomly selected from the set *SpeedNorms*. Since the behavior of each vehicle corresponds to an execution trace, the simulation generates 1,500 traces.

2. The traces are labeled by the MAS objectives evaluator as described above, obtaining the data set Γ_i that is given as input to *DDNR*.

3. Given Γ_i, the *synthesis step* is performed to generate a set of revised norms $\mathcal{R}(n_i)$.

By running 100 independent simulations, we obtain therefore 100 independent data sets Γ_1–Γ_{100}, which we use to synthesise 100 *sets* $\mathcal{R}(n_1)$–$\mathcal{R}(n_{100})$ of revised norms (one per simulation). To answer **RQ1**, we compare the confusion matrices of the synthesised norms in the set $\mathcal{R}(n_i)$ with the confusion matrix of the original norm n_i w.r.t. data set Γ_i. We report statistical results obtained in the 100 independent trials. When reporting the statistics, we use the following abbreviations: M for mean, SD for standard deviation, SE for standard error, Min for minimum value, Q1-Q3 for the three quartiles, and Max for the maximum value. These measures allow to characterize quantitatively the distributions of the number of TP, FP, TN, and FN traces (which we visualize via box plots) for the synthesised sets of revised norms. Moreover, we indicate with $\vec{n}$ the vector of the 100 original norms being revised, and with $\mathcal{R}(\vec{n})$ the vector of 100 sets of revised norms (i.e., the sets $\mathcal{R}(n_1)$–$\mathcal{R}(n_{100})$), one set per norm in $\vec{n}$. Finally, in the tables in the rest of the section, we highlight in gray the values that we mention in the text.

We repeat the above for the three types of revision (weakening, strengthening and alteration) as per Section 3.2. We make the following hypotheses about the sets $\mathcal{A}(\vec{n})$, $\mathcal{W}(\vec{n})$ and $\mathcal{S}(\vec{n})$, respectively for alteration, weakening and strengthening, which help us determining to what extent the norms synthesised with algorithm synthesis differ from the original norms w.r.t. TP, TN, FP and FN traces (**RQ1**):

H1.1 The sets of altered norms in $\mathcal{A}(\vec{n})$ are composed of norms with more TP and TN traces and fewer FP and FN traces, compared to the original norms.

H1.2 The sets of weakened norms in $\mathcal{W}(\vec{n})$ are composed of norms with more compliant traces (TP and FP traces) and fewer violating traces (TN and FN traces), compared to the original norms.

H1.3 The sets of strengthened norms in $\mathcal{S}(\vec{n})$ are composed of norms with more violating traces (TN and FN traces) and fewer compliant traces (TP and FP traces), compared to the original norms.

Table 4a reports an overview of data sets Γ_1–Γ_{100}. On average, the data sets are split relatively evenly among positive and negative traces w.r.t. the MAS objectives (TP+FN traces are 52.5%, TN+FP are 47.5%), while, on average, the traces violating the original norms are about the 7% of the total (TN+FN traces). Note that the randomly selected original norms are generally too weak: while they cover relatively well the positive traces (high values of TP), in many cases they mis-classify a large part of the negative traces (high values of FP). The average accuracy is about 54%, with a maximum value of 89% and a value below 64% for 75% of the norms (see the third quartile Q3).

Table 4b reports, instead, an overview of the norms in the 100 different sets in $\mathcal{R}(\vec{n})$, generated by the *synthesis step* for the three types of revisions. In the following (see Figure 6), we study how the norms in these sets differ from the original norms described in Table 4a. Note that the changes in TP, FP, TN and FN reported in Figure 6 are computed by mapping each synthesised norm with its original one. For instance, the change of TP for a norm $n'_i \in \mathcal{W}(n_i)$ is computed as $\text{TP}_{n'_i} - \text{TP}_{n_i}$, with TP_{n_i} the TP obtained with the original norm n_i, and $\text{TP}_{n'_i}$ the TP obtained with norm n'_i synthesised starting from (by weakening, in this example) n_i. We use the effect size metric d_{Cohen}[12] to analyze the change in the percentage of TP, FP, TN and FN (illustrated in Figure 6 via box plots) between the original and the synthesised norms, and to understand if the effect of such a change has a statistically relevant magnitude.

In the case of weakening (Figure 6a), we observe a *large*[13] reduction of the number of FN traces. The reduction of FN traces corresponds to an increase of TP traces: those traces are now correctly classified as norm-compliant. Due to the heuristic nature of Algorithm 3, which relies on the available data and on the defined operations for making formulas more or less specific based on the given states, the revision also exhibits a large *side effect* of reducing TN traces with a corresponding increase of FP traces. We do not see any negative change for the norm-compliant traces, nor any positive change for the norm-violating traces. This confirms hypothesis **H1.2** and validates the correctness of the revisions: weakening does not affect negatively the norm-compliant traces.

Analogously, in the case of strengthening (Figure 6b), **HP1.3** is confirmed, since the norm-violating traces are not affected negatively while we observe a large decrease in the number of FP traces. Also in this case, the revision exhibits a large side effect of reducing

[12] The effect size is a statistical measure that describes the strength of a phenomenon, by computing the difference between two groups of measurements in terms of their common standard deviation (Cohen, 2008). In particular, we use use Cohen's delta (d_{Cohen}) since we compare two groups of same size with similar standard deviation.

[13] We use the terms *small*, *intermediate* and *large* according to the interpretation of the effect size d_{Cohen} (Cohen, 2008). A change is considered having *no effect* if $|d_{Cohen}| < 0.2$; *small effect* if $0.2 \leq |d_{Cohen}| < 0.5$; *intermediate effect* if $0.5 \leq |d_{Cohen}| < 0.8$; and *large effect* if $|d_{Cohen}| \geq 0.8$.

Table 4: Statistics about (a) the 100 original norms enforced and (b) the 100 sets of synthesised norms. In (b), column "Nr. norms" reports the statistics about the number of revised norms synthesised for each of the 100 initial norms, and the remaining columns report the statistics of such norms in terms of TP, FP, TN, FN and accuracy. Row *Total* indicates the total number of norms synthesises starting from the 100 initial norms.

(a)

		TP	FP	TN	FN	*acc*
Original norms $\vec{n}$	M	0.50	0.44	0.04	0.03	0.54
	SD	0.27	0.24	0.06	0.04	0.22
	Min	0.00	0.01	0.00	0.00	0.10
	Q1	0.47	0.35	0.00	0.00	0.49
	Q2	0.62	0.38	0.00	0.00	0.62
	Q3	0.64	0.43	0.06	0.06	0.64
	Max	0.89	0.89	0.24	0.16	0.89

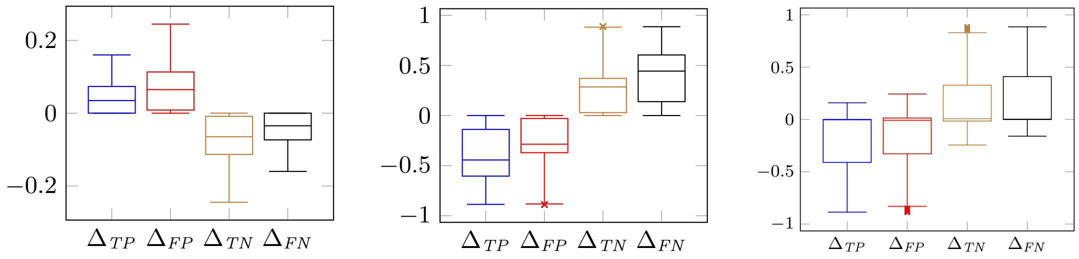
(b)

		Nr. Norms	TP	FP	TN	FN	*acc*
$\mathcal{W}(\vec{n})$	M	57.02	0.37	0.61	0.01	0.01	0.39
	SD	131.98	0.36	0.35	0.03	0.02	0.35
	Min	1.00	0.00	0.01	0.00	0.00	0.00
	Q1	1.00	0.00	0.31	0.00	0.00	0.04
	Q2	8.00	0.17	0.82	0.00	0.00	0.17
	Q3	48.00	0.66	0.96	0.01	0.01	0.67
	Max	1057.00	0.99	1.00	0.24	0.16	0.99
	Total	5702.00	-	-	-	-	-
$\mathcal{S}(\vec{n})$	M	50.86	0.15	0.12	0.28	0.45	0.43
	SD	50.77	0.21	0.18	0.26	0.28	0.25
	Min	1.00	0.00	0.00	0.00	0.00	0.01
	Q1	1.00	0.00	0.00	0.04	0.16	0.26
	Q2	35.50	0.00	0.00	0.30	0.51	0.37
	Q3	92.00	0.22	0.30	0.38	0.64	0.52
	Max	210.00	0.89	0.89	1.00	0.99	1.00
	Total	5086.00	-	-	-	-	-
$\mathcal{A}(\vec{n})$	M	218.03	0.24	0.32	0.20	0.23	0.45
	SD	229.50	0.30	0.33	0.27	0.29	0.29
	Min	1.00	0.00	0.00	0.00	0.00	0.00
	Q1	1.00	0.00	0.00	0.00	0.00	0.17
	Q2	187.00	0.12	0.29	0.06	0.04	0.43
	Q3	314.00	0.50	0.49	0.36	0.49	0.65
	Max	1409.00	0.99	1.00	1.00	0.99	1.00
	Total	21803.00	-	-	-	-	-

the number of TP traces. Note that the changes for strengthening are higher than those for weakening: the data sets include no more than 25% norm-violating traces (no more than 25% of the agents will violate the enforced norm), while up to 100% norm-compliant traces.

Since we perform weakening or strengthening revisions regardless of the actual number of FP or FN traces, in some cases no revision is needed or possible (e.g., when FP or FN traces are 0). In our simulations, this is more common for weakening (see the median values in Figure 6a closer to 0). This happens, for instance, when the enforced speed limit is very weak so that all exhibited behaviors are norm-compliant, or when, by enforcing a very strict speed limit, traffic jams are generated by compliant agents that significantly slow down, preventing also any other agent behind them to violate the norm. Similarly, if the number of FP traces is 0, no strengthening is possible/needed. This can happen, for instance, when the enforced speed limit is already reasonably aligned with the MAS objectives, and it is strict enough not to allow vehicles to speed too much, but not too strict to cause jams as described above. When no revision is possible (needed), we do not revise the original norm, resulting in no change in the classification of traces.

When performing an alteration (Figure 6c), as illustrated in Figure 3, we are not limited anymore to revisions concerning exclusively norm-compliant or norm-violating traces, and the revision can affect all types of traces. We can see this in Figure 6c by noting both an



(a) (b) (c)

		Original		Synthesised		Δ		d_{Cohen}	Interpretation
		M	SD	M	SD	M	SD		
weakening	TP	0.33	0.33	0.37	0.36	0.04	0.04	0.12	no effect
	FP	0.53	0.30	0.61	0.35	0.07	0.07	0.23	small
	TN	0.09	0.07	0.01	0.03	-0.07	0.07	-1.39	large
	FN	0.05	0.04	0.01	0.02	-0.04	0.04	-1.22	large
strengthening	TP	0.55	0.24	0.15	0.21	-0.40	0.26	-1.82	large
	FP	0.38	0.22	0.12	0.18	-0.26	0.23	-1.28	large
	TN	0.03	0.05	0.28	0.26	0.26	0.23	1.37	large
	FN	0.05	0.04	0.45	0.28	0.40	0.26	2.03	large
alteration	TP	0.43	0.32	0.24	0.30	-0.19	0.27	-0.61	intermediate
	FP	0.46	0.29	0.32	0.33	-0.14	0.25	-0.45	small
	TN	0.06	0.07	0.20	0.27	0.14	0.25	0.72	intermediate
	FN	0.05	0.04	0.23	0.29	0.19	0.27	0.91	large

(d)

Figure 6: The % *change* of TP, FP, TN, FN for the sets of norms in $\mathcal{W}(\vec{n})$, $\mathcal{S}(\vec{n})$, $\mathcal{A}(\vec{n})$ respectively obtained by (a) weakening, (b) strengthening or (c) altering the 100 original norms in $\vec{n}$, with detailed statistics: mean M and standard deviation SD (d). The mean values are obtained w.r.t. the 100 different sets of norms obtained revising the 100 original norms in the highway scenario. The change for each norm is calculated w.r.t. its original norm, and the percentage of change for each norm is calculated w.r.t. the total number of traces (1,500). d_{Cohen} and its interpretation refer to effect size as per (Cohen, 2008).

increase and a decrease in all types of traces. The sets of revised norms are more similar to the ones obtained with strengthening (at least in terms of change in the values of the confusion matrix). This shows that, in general, stricter norms are generated, in line with the fact that the original norms, as described above, are generally too weak. For this reason, the revisions mostly affect the traces that are wrongly classified as norm-compliant: the FP traces in the revised norms are reduced, even though with a small effect size. In some cases, the revisions also have the undesired side effect of increasing the number of FN traces. Therefore, **HP1.1** is only partly confirmed: it is confirmed that the set of altered norms is composed of norms with more TN and fewer FP but, due to the side effect, it is not the case that the set is composed of norms with more TP and fewer FN. However, such an effect is less significant than with strengthening, indicating that when a data set of traces includes both FP and FN traces, alteration operations allow to generate norms that are better aligned with the MAS objectives. In particular, the size of the effect on the FN traces is smaller than that of the analogous effect for strengthening.

We conclude that the results confirm the hypotheses **H1.2**, and **H1.3**, and partly confirm hypothesis **H1.1**. In particular, the results show that among the generated norms, some can have significant side effects. This motivates us to propose the use of a metric like accuracy to select among the possible norms during the *selection step*, instead of, for example, randomly extracting one norm from the synthesised set. In the following, we show that using such a metric is crucial to minimize the side effects of the revision.

4.3 Is the Accuracy of the Norms Revised With *DDNR* Higher Than the Accuracy of the Original Norms? (RQ2)

We take the sets of norms generated for **RQ1**, and analyze how the accuracy of the selected norms changes w.r.t. the original norms when the entire *DDNR* algorithm (including the *selection step*) is applied. We hypothesise that:

H2.1 The accuracy of the norms revised with *DDNR* is higher than the accuracy of the original norm in the case of weakening, strengthening and alteration operations.

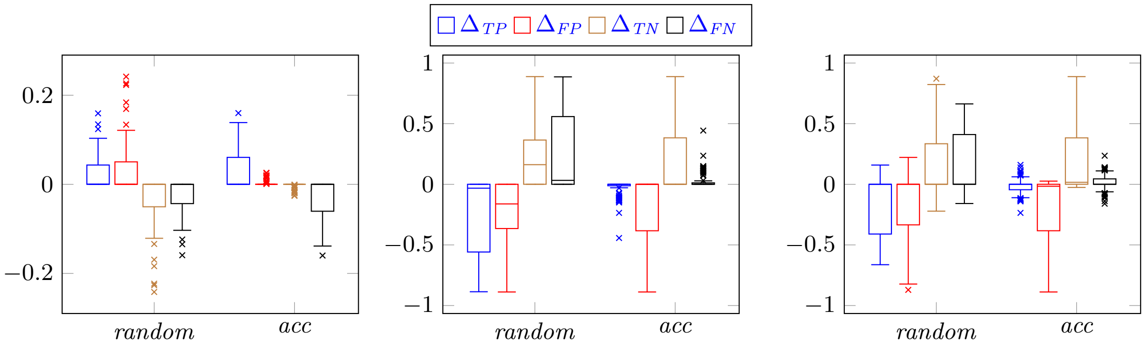
We compare the *acc* metric against a baseline unbiased metric, which we call *random*, that selects the new norm by randomly and uniformly sampling the sets produced by the synthesis step. Figure 7 illustrates such comparison via box plots. The reported values show the change of percentage of TP, TN, FP and FN traces with the 100 revised norms, obtained in the 100 trials, w.r.t. each of the 100 original norms. In Table 5, instead, we provide the detailed statistics about the selected norms. For brevity, we report only mean and standard deviation. Note that, since Table 5 concerns the statistics of the 100 norms randomly selected from the sets of synthesised norms (one from each of the 100 sets), in the case of the *random* metric the values are not identical to those in Table 4b which, instead, refers to all the norms contained in the 100 sets.

Table 5: Mean M and standard deviation SD of the 100 revised norms. The values in the TP, FP, TN and FN columns are percentages w.r.t. the number of traces in the data set.

	metric	TP	FP	TN	FN	accuracy
weakening	*random*	0.52 ± 0.29	0.47 ± 0.28	0.01 ± 0.01	0.01 ± 0.01	0.52 ± 0.28
	acc	0.52 ± 0.29	0.44 ± 0.24	0.04 ± 0.06	0.00 ± 0.01	0.56 ± 0.24
strengthening	*random*	0.24 ± 0.27	0.21 ± 0.20	0.27 ± 0.31	0.28 ± 0.32	0.51 ± 0.25
	acc	0.47 ± 0.28	0.19 ± 0.17	0.28 ± 0.39	0.06 ± 0.08	0.75 ± 0.15
alteration	*random*	0.33 ± 0.29	0.29 ± 0.27	0.18 ± 0.25	0.20 ± 0.27	0.51 ± 0.23
	acc	0.48 ± 0.29	0.18 ± 0.18	0.30 ± 0.39	0.04 ± 0.07	0.78 ± 0.15

Weakening. Figure 7 shows that, while with *random* the percentage of FP traces increases, on average, of about 3%, with *acc* the increase is significantly lower (on average 0)[14]. In terms of change in the FN and TP traces, instead, *random* and *acc* perform analogously. By only improving the FP and TP, *acc* leads to norms that are both more accurate than those selected with *random* (compare the accuracy in Table 5) and also more accu-

[14] A Wilcoxon-Signed Rank Test indicates a significant difference between the change of the percentage FP traces with *random* and with *acc*; $z = -5.47, p < .00001$, with a large effect size $d_{Cohen} = -0.832$



(a) weakening (b) strengthening (c) alteration

metric		weakening	strengthening	alteration
random	Δ_{TP}	0.02 ± 0.04	-0.25 ± 0.29	-0.17 ± 0.24
	Δ_{FP}	0.03 ± 0.05	-0.23 ± 0.26	-0.15 ± 0.22
	Δ_{TN}	-0.03 ± 0.05	0.23 ± 0.26	0.15 ± 0.22
	Δ_{FN}	-0.02 ± 0.04	0.25 ± 0.29	0.17 ± 0.24
acc	Δ_{TP}	0.02 ± 0.04	-0.03 ± 0.07	-0.02 ± 0.07
	Δ_{FP}	0.00 ± 0.01	-0.24 ± 0.34	-0.26 ± 0.33
	Δ_{TN}	0.00 ± 0.01	0.24 ± 0.34	0.26 ± 0.33
	Δ_{FN}	-0.02 ± 0.04	0.03 ± 0.07	0.02 ± 0.07

(d)

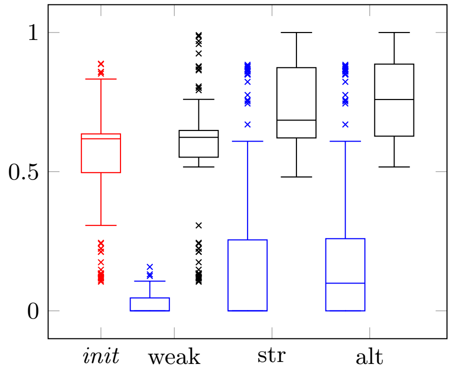
Figure 7: Comparison of the *change* of the % of TP, FP, TN and FN after (a) weakening, (b) strengthening or (c) altering a norm with *DDNR*, and (d) the corresponding values.

rate than the original norms (56% accuracy compared to the original 54%). This confirms hypothesis **HP2.1** in the case of weakening of one norm.

Strengthening. Figure 7 shows that, while both *random* and *acc* lead to a desired reduction of the number of FP (in both cases of about 23–24%), *acc* has a side effect on the FN about 8 times smaller than that caused by *random*, limiting the side effect on the FN to an average 3% change, compared to the 25% change obtained with *random*. As a consequence, the norms selected by *acc* are also more accurate than those selected by *random* (in Table 5, we see an average accuracy of 75% with *acc*, and of 51% with *random*). This confirms hypothesis **HP2.1** in the case of strengthening of one norm.

Alteration. Consider row *alteration* in Table 5. *acc* significantly improves in all classes compared to *random*, selecting norms with significantly more TN traces (and consequently fewer FP traces) than the original norms (with a large effect $d_{Cohen} = 0.932$), and with a side effect on FN and TP (see highlighted cells in column alteration of Figure 7) which has a size about 5 times smaller than the effect with *random* (also, the effect size between the number of FN of the original norms and the number with the revised norms is $d_{Cohen} = 0.175$ for *acc* and $d_{Cohen} = 0.881$ for *random*). *acc* effectively selected norms with higher accuracy than the original ones, with an average accuracy of 78% (see last cell in Table 5) compared to the 54% accuracy (shown in Table 4a) of the original norms. This confirms hypothesis

HP2.1 also in the case of alteration of one norm.



(a)

	Original
M	0.54
SD	0.22
Min	0.10
Q1	0.49
Q2	0.62
Q3	0.64
Max	0.89

(b)

		Δ	*Final*
weakening	M	0.02	0.56
	SD	0.04	0.24
	Min	0	0.10
	Q1	0	0.55
	Q2	0	0.62
	Q3	0.05	0.65
	Max	0.16	0.99
strengthening	M	0.21	0.75
	SD	0.33	0.15
	Min	0	0.48
	Q1	0	0.62
	Q2	0	0.68
	Q3	0.26	0.87
	Max	0.88	1.00
alteration	M	0.24	0.78
	SD	0.32	0.15
	Min	0	0.52
	Q1	0	0.63
	Q2	0.10	0.76
	Q3	0.26	0.89
	Max	0.88	1.00

(c)

Figure 8: Sub-figure (a) shows the original accuracy in red, the accuracy change in blue, and the resulting accuracy after revision in black; sub-figure (b) reports the original accuracy from Table 4a; sub-figure (c) illustrates the Δ and the resulting accuracy after the revision for with the three types of revision.

Figure 8 provides a more detailed comparison of the accuracy of the original norms with the accuracy of the revised norms when using *acc* as a metric, and shows the accuracy change when performing weakening, strengthening or alteration. As reported earlier in Table 4a, the accuracy of the original norms is, on average, about 54%, with 75% of the norms having an accuracy below 64%, while the maximum accuracy is 89%. Both weakening and strengthening operations in some cases affect the accuracy positively and in other cases negatively. On average, however, all operations led to norms with significantly higher accuracy than the original norms, confirming **HP2.1** for all types of revisions of one norm. Wilcoxon-Signed Rank Tests comparing the accuracy of the original norms with the accuracy of the altered norms indicate a significant difference, with $z = -8.6818, p < .00001$ for weakening, $z = -3.9902, p = .00006$ for strengthening, and $z = s - 3.593, p = .00034$ for alteration. Due to the low number of negative traces, the improvement with weakening has a smaller magnitude, with a negligible effect size $d_{Cohen} = 0.087$. In the case of strengthening, instead, the average improvement of accuracy is around 20%, with a large effect size $d_{Cohen} = 1.115$. Finally, with alteration, the average accuracy improves even more, with a large improvement of around 24% ($d_{Cohen} = 1.275$). Note that half of the new norms have an accuracy higher than 76%, and 25% of the norms have an accuracy higher than 89%.

Multiple Norms

We experiment with the highway scenario enforcing *two norms* instead of one, in order to answer **RQ2** also for the case of multiple norms. We focus only on alteration here for brevity. The results on weakening and strengthening, which are analogous to the case of one norm (i.e., the accuracy of both the weakened and strengthened norms is higher than the original norms), are included in the online appendix (Dell'Anna et al., 2022a).

Method. Similar to the case of a single norm, we adopt the following experimental method. We run and report results for 100 independent simulations. For each simulation, we perform the following three steps. (1) We run the *i-th* simulation until a data set of 1,500 traces is generated under the enforcement of *two norms* (as opposed to one, in the case of one norm) randomly selected from the sets *SpeedNorms* and *DistanceNorms*. (2) The MAS objectives evaluator labels the traces as per **RQ1**, producing the data set Γ_i. (3) Given Γ_i, we run *DDNR*. Differently from the case of one norm, we repeat this step for two different experiments—the subject of our analysis—in order to investigate two different strategies to evaluate and select multiple norms: (*independent*) we select, by using the *acc* metric, each norm from its corresponding set, independently form the other norm; (*combined*) we compare all combinations of norms from the two sets, and we select the two revised norms by using the multi-label accuracy metric *ml-acc* defined in Equation 2.

Table 6 compares the original norms with the revised ones, for the two experiments of independent and combined revision of the norms. In column *independent*, the values concern the average *acc* of the norms, and in column *combined*, the values concern the value of *ml-acc*. By comparing the mean values of the original and revised norms, we observe that in both experiments the alignment with the MAS objectives (i.e., the average accuracy of the two norms in *independent*, and the multi-label accuracy in *combined*) significantly[15] improves after the revision of the norms. This effect is large for both the independent ($d_{Cohen} = 1.205$) and the combined ($d_{Cohen} = 0.892$) revisions, and confirms **HP2.1** also for the revision of multiple norms.

Table 6: Statistics about the accuracy (*acc*) and multi-label accuracy (*ml-acc*) for the original and revised (altered) norms in the cases of *independent* and *combined* selections.

Experiment	*independent*		*combined*	
Metric	*acc*		*ml-acc*	
Norms	Original	Revised	Original	Revised
M	0.50	0.74	0.47	0.68
SD	0.25	0.13	0.28	0.18
Min	0.06	0.50	0.00	0.00
Q1	0.25	0.63	0.16	0.61
Q2	0.61	0.72	0.61	0.63
Q3	0.63	0.82	0.63	0.77
Max	0.93	1.00	0.93	0.99

[15] We conducted two Wilcoxon-Signed Rank tests to compare the accuracy of the original and revised norms. In both cases we identify a significant difference, For the *independent* revision we have $z = -7.9607, p < .00001$. For the *combined* revision we have $z = -7.3738, p < .00001$.

In Section 3.4, we discussed the difference between an independent and a combined revision of the norms in terms of partly and fully correctly classified traces. We argued that independent revisions are more suitable for cases when it is less important for the traces to be classified correctly by all norms, while combined revisions are more suitable when it is important to have fully correct traces. We briefly illustrate this concept in our experiment. Figure 9 reports three confusion matrices: (a) one for the original norms of our experiment, (b) one for the revised norms obtained in the *independent* experiment, and (c) one for the revised norms obtained in the *combined* experiment. Note that the values in the confusion matrices are the mean values for the 100 different norms (e.g., the top left value in Figure 9a, is the mean number of *Positive Fully Correct* traces in the 100 original norms). For the sake of readability, in this last qualitative discussion, we omit from the figure the standard error of the mean values.

$$\begin{array}{cc|cccc|} & & \multicolumn{2}{c}{n_s\ \textit{compliant}} & \multicolumn{2}{c}{n_s\ \textit{violating}} \\ & & n_d\ \textit{compliant} & n_d\ \textit{violating} & n_d\ \textit{compliant} & n_d\ \textit{violating} \\ \text{objectives} & \textit{positive} & 689.8 & 8.9 & 34.08 & 0.08 \\ & \textit{negative} & 687.95 & 19.63 & 59.58 & 1.41 \end{array}$$

(a)

$$\begin{array}{cc|cccc|} & & \multicolumn{2}{c}{n_s\ \textit{compliant}} & \multicolumn{2}{c}{n_s\ \textit{violating}} \\ & & n_d\ \textit{compliant} & n_d\ \textit{violating} & n_d\ \textit{compliant} & n_d\ \textit{violating} \\ \text{objectives} & \textit{positive} & 640.52 & 11.24 & 64.37 & 16.73 \\ & \textit{negative} & 207.39 & 29.53 & 220.57 & 311.08 \end{array}$$

(b)

$$\begin{array}{cc|cccc|} & & \multicolumn{2}{c}{n_s\ \textit{compliant}} & \multicolumn{2}{c}{n_s\ \textit{violating}} \\ & & n_d\ \textit{compliant} & n_d\ \textit{violating} & n_d\ \textit{compliant} & n_d\ \textit{violating} \\ \text{objectives} & \textit{positive} & 694.55 & 0.45 & 21.47 & 16.39 \\ & \textit{negative} & 355.45 & 2.36 & 101.28 & 309.48 \end{array}$$

(c)

Figure 9: Confusion matrices for the classification of the 1,500 traces with the (a) the original norms, (ii) the revised norms obtained in the *independent* experiment, and (iii) the revised norms obtained in the *combined* experiment. The values in the matrices are the mean values for the 100 different norms.

Figure 9a highlights how the original norms, on average, fully correctly classify almost all the positive traces (top left value), while they fully wrongly classify almost all the negative traces (bottom left value). This shows that the original norms (both of them) are generally too weak and too many negative behaviors are classified as norm-compliant. Comparing the matrices in Figure 9b and Figure 9c, we find a confirmation of the above discussion. Both the independent and combined approaches improve the alignment of the norms with the MAS objectives, reducing the fully wrongly classified negative traces (bottom left value) and improving the fully correctly classified negative traces (bottom right value). With the *independent* revision, however, we observe a lower number of fully correctly classified positive traces (top left) and a higher number of partly correctly classified traces (the four central values). With the *combined* revision, instead, there is a higher number of fully correctly classified positive traces, and, while the number of fully wrongly classified negative traces is higher than with the independent approach, the revised norms have lower number of partly correctly classified traces (compare the four central values of the two matrices).

The results of this section show that the accuracy of the norms improves after revising them via *DDNR*, and confirms hypothesis **H2.1**. In all cases, as we have seen comparing Figure 7 with Figure 6, accuracy provides a useful filter to select, from the set of possible norms generated in the *synthesis step*, those that are better aligned with the MAS objectives, while discarding the ones with strong side effects. Similar results are obtained for multiple norms. These results show that the norms revised with *DDNR* are significantly better aligned with the MAS objectives on the given data sets than the original ones. Even though the accuracy improves significantly, in some cases it is not perfect. We will discuss further this aspect in Section 5.

4.4 How Does *DDNR* Generalize to Previously Unseen Traces? (RQ3)

So far, we did not discuss the case when the data set Γ is not a complete sample of all possible traces. To answer **RQ3**, which concerns the generality of the norms obtained with *DDNR*, we perform two experiments. In the first experiment (*75-25 splitting*), we use a standard (machine learning) splitting technique: we take the 100 data sets obtained for **RQ1** and we split each of them in a training and a test set, composed of 75% and 25% of the traces in the data set, respectively. In the second experiment (*Independent test set*), we use the full data sets as training sets, and a new set of traces (obtained by running the highway simulation with no norm enforced) as test set. In both experiments, we apply *DDNR* on the training set, as done per **RQ2**. This time, however, we compare the accuracy of the original norm and the revised one on the test set. We hypothesise that:

H3.1. The accuracy of the revised norms is higher than the accuracy of the original norms on previously unseen traces.

We focus only on alteration again for brevity, and include analogous results for weakening and strengthening in the online appendix. Table 7 reports the results for both experiments.

75-25% splitting. First, on the training set the revised norms have significantly higher accuracy than the original ones, in line with the results obtained for **RQ2** (compare the values here with the values for alteration in Figure 8). Analogous results are obtained also on the test set, composed of previously unseen traces: while the original norms have an average accuracy of about 53%, the revised norms have an accuracy of about 78%. This confirms hypothesis **HP3.1** in the case of 75-25% splitting. Comparing the accuracy of the revised norms on the training and test set, we notice that the accuracy is identical on average, showing that the revised norms seem to generalize well also to previously unseen traces. Since we split the original data sets, however, the test sets traces (although unseen

Table 7: Statistics about the accuracy of the original and revised (altered) norms on the training and test sets. In *75-25 splitting*, the training and test sets are composed, respectively, by 75% and 25% of the traces in the original data sets. In *Independent test set*, the training set is the full set of traces in the original data sets, and the test set is an independent set of traces obtained running the simulation with no norm enforced.

	75-25 splitting				*Independent test set*			
	Training set		Test set		Training set		Test set	
Norms	Original	Revised	Original	Revised	Original	Revised	Original	Revised
M	0.54	0.78	0.53	0.78	0.54	0.78	0.50	0.59
SD	0.22	0.15	0.22	0.15	0.22	0.15	0.25	0.21
Min	0.11	0.52	0.08	0.49	0.11	0.51	0.11	0.12
Q1	0.48	0.63	0.50	0.63	0.49	0.63	0.35	0.38
Q2	0.62	0.76	0.61	0.77	0.62	0.75	0.61	0.62
Q3	0.64	0.89	0.64	0.90	0.63	0.89	0.64	0.75
Max	0.89	1.00	0.87	1.00	0.89	1.00	0.90	0.90

during the revision process) are not completely independent from the traces composing the training sets, for they are both obtained by monitoring the behavior of the agents under the enforcement of the same original norms.

Independent test set. When we compare the difference in the accuracy of the revised norms on the training and test set, we observe that the accuracy is about 19% lower on the test set ($d_{Cohen} = 1.041$). This shows a certain, expected, degree of overfitting to the training set[16]. However, the revised norms are still better aligned than the original norms also on the previously unseen traces composing the test set: while the original norms have an average accuracy of about 50%, the accuracy of the revised norms is about 9% higher (corresponding to a small increase $d_{Cohen} = 0.39$). This confirms hypothesis **HP3.1** also in the case of independent data.

The results of this section show that the revised norms are better aligned with the MAS objectives than the original norms, and they also appear to generalize better than the original norms on previously unseen traces, confirming the hypothesis **H3.1**.

5. Discussion and Limitations

In this section, we discuss some of the most important aspects that characterize *DDNR* and some of its limitations.

Perfect accuracy. From the experiments reported in Section 4 (e.g., see Table 7), we note that even though on average the accuracy of the revised norms is higher than the accuracy of the original ones, *DDNR* sometimes fails to obtain perfect accuracy. This is due to two reasons: (i) the impossibility of approximating perfectly the MAS objectives by means of conditional norms, (ii) the types of norms that can be synthesised by means of our revision operations. We briefly discuss these two reasons.

[16] We do not discuss them here, but in order to mitigate the effect of overfitting, different techniques, such as cross-validation techniques (Kohavi, 1995), can be employed while the training of a model (in our case, during the revision process).

One or more conditional norms may not be sufficiently expressive to perfectly characterize the MAS objectives. In our simple highway scenario, for example, we are trying to align (a) one norm concerning the speed limit of the vehicles in the highway section with (b) MAS objectives concerning a combination of travel time and CO_2 emissions. Regulating only the speed of the cars (and doing it only with one norm), obviously, may not always be sufficient to achieve any possible objective. For example, the CO_2 emitted by a vehicle does not only depend on its speed but also on its acceleration. Norms regulating only the speed of the vehicles may not be sufficient, therefore, to fully characterize (and achieve) the MAS objectives. In this paper, we discussed how to revise a given set of norms w.r.t. a certain data set of traces. If the considered types of norms (e.g., speed related norms) are not expressive enough to fully characterize the MAS objectives (e.g., objectives related to both CO_2 and throughput at the same time), it is possible that no norm expressible in the given language can achieve perfect accuracy on a data set of traces.

The second reason why perfect alignment is not always reached concerns the types of norms that can be synthesised w.r.t. a given data set. *DDNR* heuristically determines approximate norm revisions as a more practical solution than exhaustively searching the space of all possible norms, which is intractable (Dell'Anna et al., 2022b). To do so, as seen in Section 3, Algorithms MoreSpec and LessSpec rely on the following two aspects.

(A) The propositions that are true in the states of traces in the data set (e.g., the states *CS* or *CPS* defined in Section 3). If the data set is not informative enough, the revised norms produced by our algorithms, which rely on the given data, may not contain a new norm that is perfectly aligned with the MAS objectives. Consider an example where the data set contains no norm-violating traces (e.g., if the data set is generated by monitoring the enforcement of a very strict speed limit that forces all agents to go very slowly). In such a case, the data set only provides evidence about TP and FP traces. Moreover, if a traffic jam is caused by all the agents going very slowly, most likely all the traces will be labeled as negative w.r.t. the MAS objectives, thus the data set would actually provide evidence only about FP traces. With this type of data set, *DDNR* will likely result in norms that are not better aligned with the MAS objectives than the original ones. In the example, by considering only data about norm-compliant traces, *DDNR* will be able to synthesise only stricter norms. The MAS designer might be required instead to weaken the speed limit, thereby allowing more vehicles to speed up. Generally, in order to retrieve sufficient information to produce meaningful revisions, it might be necessary to collect data about the behavior of the agents under the enforcement of different (and possibly no) norms. Potentially, data collection can be performed in an iterative and continuous fashion (i.e., by iteratively applying *DDNR* after collecting data from the enforcement of the revised norms), and the decision about how to revise a norm in a given iteration (e.g., whether to weaken it, alter it or strengthen it) might follow from strategies that reflect the current state of the system w.r.t. the achievement of the MAS objectives (e.g., in line with the considerations reported by Dell'Anna et al., 2020). These aspects, which are important for an automated run-time revision of norms, belong to our future work.

(B) The function buildConj. As mentioned in Section 4.1, in our experiments, we bounded the number of propositions to be considered during the *synthesis step*. In particular, we selected a limited number of higher or lower speeds for the prohibited state, and positions for the condition and deadline. Consider, as an example, an original norm prohibiting vehicles from having a speed higher than 10 km/h. When generating more specific speed limits, we limited the maximum number of more specific speed limits to 8 (for example the speeds 11, 15, 16, 20, ..., 24). For instance, if the MAS objectives are not achieved when a vehicle is speeding over 40 km/h, but they are achieved when a vehicle is having a speed of 30 km/h, due to the limitations imposed to the function buildConj aimed at bounding the complexity of the algorithm, we may not generate a new norm that perfectly distinguishes the negative and positive traces. One way to try to mitigate this effect is by revising one norm multiple times. By doing so, we may be able to correct some additional (possibly new, in case of alteration) errors that could not be captured only with one revision, like in the example above. Starting from an original norm n, we can alter it into a new norm n' and then we can alter n' into a further alteration n'', etc.. In our experiments, however, this strategy did not lead to any further improvement of accuracy.

Labeling of traces. We assumed that it is possible to determine for every trace a label based on whether it contributes or not to the achievement of the MAS objectives. While this assumption is realistic in a number of contexts and types of MAS objectives, as discussed in Section 2, the labeling of an individual trace w.r.t. the MAS objectives is not always deterministic. For example, if the MAS objective is to avoid traffic jams, the presence of a traffic jam, which is a property of the whole system, cannot be determined for an individual trace. In some cases, therefore, the MAS objectives may concern aggregate data (i.e., sets of behaviors all together), or their evaluation can be obtained only with a lower frequency than the rate at which the traces are obtained (e.g., if the MAS objectives concern user feedback that is obtained every few weeks or months). While this is a current limitation of our approach, we believe that *DDNR* can be used also for traces that are non-deterministically labeled by the MAS objectives evaluator, where different instances of the same trace can have a different label and belong to both the positive and the negative classes. This does not affect the revision operations, as the operations do not depend on the MAS objectives (e.g., to make the condition of a norm more specific, we consider the traces that violate the norm, regardless of their classification according to the MAS objectives). Once the set of possible norms is synthesized, the final norm is chosen based on statistical measures, which can be calculated also on a data set that is non-deterministically labeled, as it is typically done in automated classification.

We leave experimental studies about non-deterministic labeling for future work. As an illustrative example, however, consider a data set containing, among other traces, ten instances of a trace t describing a certain behavior of a truck in the highway. Suppose that such a trace is labeled nine times as positive by the MAS objectives evaluator and one time as negative. This could happen, for instance, if the MAS objective is to avoid traffic jams, and the behavior of the truck described by t was observed only one time out of 10 during a traffic jam, i.e., the behavior does not highly correlate with a traffic jam. In the *synthesis step*, t is used to synthesize candidate revisions of a norm n. In the *selection step*, if one of the candidate revisions, let it be n', prohibits trace t, n' will result in one TN, but also in nine FN. Norm n' will therefore have a low accuracy (at least for what concerns trace t, since it prohibits a behavior that is not highly correlated with traffic jams), and other norms may be preferred to n', thereby taking into account the rationale behind the non-deterministic labeling.

Norms and language. The current proposal only supports conditional prohibitions with deadlines, with components expressed as propositional formulas in DNF. While these norms allow to describe *temporal* patterns of behaviors, their expressiveness might not be sufficient in every domain to characterized the behaviors to regulate. The general methodology for data-driven norm revision described in this paper (i.e., first synthesise then select via statistical measures w.r.t. the available data) is applicable regardless of the type of norms. However, the synthesis step of *DDNR* depends on the specific type of norms considered. To support different types of norms, changes are necessary to Algorithms 1 and 2, which assume formulas expressed in DNF with no negation, and to Algorithm 3, which extracts states from the data set of traces based on the semantics of conditional norms (e.g., see the patterns of states described in Figure 3a). Extending *DDNR* to conditional *obligations* should be straightforward, due to their duality with prohibitions (Alechina et al., 2014). Different synthesis operators should be defined, however, for other types of norms (e.g., prohibitions or obligations defined in more complex modal logics). Defining similar operations for more complex languages than that used in this paper is subject of future research.

Similarly, another limitation of the norms that we considered is that they do not distinguish multiple violations in a trace, i.e., a trace is either classified as compliant (when the norm is never violated in the trace) or violating (if at least one violation occurs). This aspect might be limiting in some contexts. Moreover, we assume that a trace can be given a *boolean* evaluation also w.r.t. the MAS objectives. Generally speaking, however, a trace can have degrees of alignment with the MAS objectives. To support a more fine-graded evaluation of traces, it is necessary to consider a number of aspects that we did not address in this paper, including the distinction between multiple violations of a norm in a trace, the attribution of a *compliance score* (or *violation score*) to the traces, and the selection of the revised norms based on metrics, different from accuracy, that are appropriate to non-binary classification problems (e.g., the Mean Squared Error).

6. Related Work

In the MASs literature, norms have been proposed as a way to regulate the behavior of the agents in order to achieve system-level properties without limiting the autonomy of the agents (Alechina et al., 2015; Vázquez-Salceda et al., 2008).

A variety of formal approaches have been proposed for proving the correctness of normative systems by checking formulas that describe properties of a MAS, such as liveness or safety, against a model of the MAS (Alechina et al., 2013; Knobbout and Dastani, 2012; Wooldridge and van der Hoek, 2005; Ågotnes et al., 2007). These solutions are extremely useful for the design-time construction of robust normative MASs. However, they rely on the availability of a formal modeling and representation of the MAS, often feasible only for simple and relatively static systems. Shoham and Tennenholtz (1995) (see also Fitoussi and Tennenholtz, 2000), for example, consider the problem of synthesising a *social law* that constrains the behavior of the agents in a MAS so as to ensure that agents in a *focal* state are always able to reach another focal state no matter what the other agents in the system do. They show that synthesising a useful social law is NP-complete. van der Hoek et al. (2007) recast the problem of synthesising a social law as an ATL model checking problem. The authors show that the problem of whether there exists a social law satisfying an objective expressed as an arbitrary ATL formula (feasibility) is NP-complete, while for objectives expressed as propositional formulae, feasibility (and synthesis) is decidable in polynomial time. Model checking, moreover, cannot fully cope with the runtime unpredictability of the system that stems from the autonomy and heterogeneity of the agents, like in the case of open MAS, where the MAS designers have no complete knowledge of the (internals of the) agents that will join the system (Artikis and Pitt, 2001). Fixed normative systems, therefore, cannot generally perform well in any kind of runtime situation (Miralles et al., 2013). As a consequence, the synthesis and revision of norms have gained attention over the years, and it has been approached from different points of view (Sims et al., 2008; Kota et al., 2008; Savarimuthu and Cranefield, 2011; Airiau et al., 2014).

In the following, we briefly overview work in the literature that most relates to ours, highlighting the major distinctions of our proposal from the existing solutions.

Alechina et al. (2014), introduce the concept of norm approximation in the context of imperfect monitors, i.e. monitors that cannot perfectly detect violations of a norm, but can detect violations of an imperfect representation of the norm. The authors illustrate how to synthesise a norm to approximate an original norm in order to maximize the number of violations that an imperfect monitor can detect. Although presented with a different goal in mind, this work inspires our paper. In this work, however, we assume perfectly monitorable norms, and we focus on the synthesis of norms that are better aligned with the MAS objectives with respect to execution traces by using a data-driven approach.

Miralles et al. (2013) present a framework for the adaptation of MAS regulations at runtime similar to ours. They consider norms expressed via norm patterns (i.e., IF-THEN rules associated with constraints on the operators and on the values that the norm components can take). The authors describe an adaptation mechanism based on Case-Based Reasoning. The authors propose a framework where norm adaptation is performed at runtime first individually by a number of assistant agents and then, via a voting mechanism, a final adaptation is approved. The decision on how to adapt norms is taken based on similar previously seen cases. On the same lines, Dell'Anna et al. (2020) propose a framework for the runtime selection of alternative norms based on Bayesian Networks trained with runtime data to capture the corelation between the enforcement of norms and the achievement of systems objectives. The authors illustrate how to revise the sanctions of norms based on the learned knowledge and on information about the agents preferences.

These approaches, which tackle a problem analogous to ours, make a number of assumptions about the agents participating in the MAS, and about their internals. In particular, Dell'Anna et al. (2020) assume knowledge about the preferences of the agents, while Miralles et al. (2013) assume that some agents with specific reasoning, monitoring, and communication capabilities can be placed in the MAS. In our work, we relax such assumptions: our proposal treats agents in the MAS as black boxes, and *DDNR* exclusively relies on collected execution data.

Corapi et al. (2011) and Athakravi et al. (2012) discuss the application of Inductive Logic Programming (ILP) (Lavrac and Dzeroski, 1994) to norm synthesis and norm revision. In their work, the desired properties of the system are described through use cases (event traces associated to a desired outcome state). Given the use cases, the authors propose to use ILP to revise the current norms so to satisfy the use-cases. In such approach, norms and desired outcome are strictly coupled: the desired outcomes of execution traces are expressed *in the same language* of the norms and, therefore, are directly enforceable. In our approach we consider a type of desired MAS objectives that cannot be directly enforced, and we use norms as a means to achieve such objectives (e.g., a speed limit norm is a means to achieve vehicles' safety, but it is not possible to directly enforce safety on vehicles: "no accidents should occur" is not directly enforceable on drivers). In our work, the only knowledge of the MAS objectives available to the revision mechanism is a given boolean labeling of the execution traces. The causal relation between norms and MAS objectives is not given. Because we do not assume that the underlying causal structure of the domain is known to our revision mechanism, we are unable to generate provably correct norm revisions as in ILP-based approaches like (Corapi et al., 2011) and related ones (Katzouris et al., 2015; Muggleton et al., 2014). This is why we use statistical analysis to drive the revision of norms. Our approach and ILP-based approaches, can therefore be seen as representing different trade-offs between the amount of background knowledge assumed about the possible causes of norm violations, and the guarantees that can be given regarding a particular (candidate) revision.

Lion (Morales et al., 2015) is an algorithm for the synthesis of liberal normative systems, i.e., an algorithm that synthesises norms trying to set as few constraints as possible on the agents' actions. To guide the synthesis process, the authors make use of a normative network: a graph structure that characterizes the generalization relationship between different norms. They use such graph to synthesise more general, that is more liberal, norms when possible. The norms synthesised by Lion are so-called action-based norms, which prohibit agents from performing actions in certain states (Alechina et al., 2018). In our work, we focus on the problem of revising conditional norms with deadlines, which are behavior-based, or path-based, norms, prohibiting agents from exhibiting certain behaviors. Furthermore, differently from Lion, where the norms are synthesised so to achieve certain properties of the normative systems (such as its liberality), our norm revision is meant to align the enforced norms with MAS objectives, which are properties that are desired from the *behavior* of the MAS. We consider the liberality aspects of the norms an interesting possible extension of our work that could be integrated as a criterion when selecting a new norm among the possible revisions in the *selection step*.

The concept of generalization of a norm described by Morales et al. (2015) also relates to our operations of weakening and strengthening of a norm. Weakening a norm generates more general norms, while strengthening generates more specialized ones. Similar operations have been described also in the software engineering literature, for example by Kafali et al. (2017) who define design patterns for the iterative revision and verification of a specification, a work which was later extended in DESEN (Kafali et al., 2020) to encompass also socio-technical systems. We propose a data-driven approach to perform similar operations for conditional norms with deadlines.

Game theoretic concepts have been employed to guide norm synthesis. Perelli (2019), for example, show that the synthesis of dynamic norms for LTL objectives and Nash equilibria is 2EXPTIME-complete when considering the existence of a Nash equilibrium satisfying the objective, and in 3EXPTIME for enforcing all Nash equilibria to satisfy the objective. Morales et al. (2017) introduce a control loop which includes game recognition, payoff learning, and norm replication. Unlike us, they focus on the goals of the individual agents. In our setting, instead, we concentrate on objectives of the MAS, which may differ and conflict with the goals of the individual agents. Similarly, Mahmoud et al. (2018) propose an algorithm for mining regulative norms that identifies recommendations, obligations, and prohibitions by analyzing events that trigger rewards and penalties. They focus on agents joining an open MAS who have to learn the unstated norms; we, instead, study how to alter existing norms from the point of view of a centralized authority that is exogenous to the agents. Bulling and Dastani (2016) have used concurrent game structures to illustrate how the enforcement of norms can change agent behavior via regimentation and sanctions. Their work shows the high computational complexity of reaching a Nash equilibrium mapping, thereby motivating our data-driven (approximate) revision mechanism. Related to game theoretical settings, where typically norm regimentation is considered, Christelis and Rovatsos (2009) devise algorithms that introduce prohibitions in a MAS by setting preconditions to the actions the agents can perform in a regimentation setting. In our work, differently from the works mentioned above, we do not explicitly consider a game theoretical setting and we do not assume that regimentation is available.

Finally, our work is influenced by research on norm change, including logics for norm change (Knobbout et al., 2016; Aucher et al., 2009), the study of the legal effects of norm change, analyzed and formalized by Governatori and Rotolo (2010), and the contextualization of norms (Jiang et al., 2012), which studies how to refine norms to make them suitable for specific contexts. In our framework, this corresponds to modifying the detachment condition and the deadline of the norms.

7. Conclusions and Future Work

We investigated the problem of norm revision in contexts where the internals of the agents in a MAS are unknown and where norms are expressed in a different language from that of the MAS objectives that they intend to bring about. In such setting, explicit knowledge about the relationship between the enforced norms, the agents' behavior and the MAS objectives is not given, and a norm revision mechanism can solely rely on the monitored system's execution. We presented results regarding the revision of conditional norms (prohibitions) with deadlines w.r.t. a set of observed traces representing the behavior of the agents in the MAS. The traces are partitioned into positive and negative ones, depending on whether each helps or hurts MAS objectives. Besides a boolean evaluation, the revision mechanism possesses no information about the relationship between a trace and the objectives. We proposed *DDNR* (Data-Driven Norm Revision): a practical heuristic approach to obtain approximate revisions of the conditional norms.

DDNR consists of two steps: the *synthesis step* and the *selection step*. The synthesis step generates a set of candidate new norms by revising the original norms based on the given data set of execution traces. The selection step selects the final norms from the synthesised set. In doing so, a norm is interpreted as a binary classifier distinguishing norm-compliant and norm-violating execution traces. Statistical metrics, such as accuracy, are used to evaluate and select the best norm among the possible candidates w.r.t. the classification of the traces provided by the MAS objectives. We applied *DDNR* to a traffic simulation, and we studied the accuracy of the revised norms. Results show that the revised norms are significantly more accurate (aligned with the MAS objectives) than the original norms, exhibiting an average improvement of accuracy on the given data set of traces of about 30% and an average improvement of accuracy of about 13% on unseen traces.

In future work, we intend to embed *DDNR* in the runtime supervision framework presented in earlier work (Dell'Anna et al., 2019, 2020) that continuously monitors the system's execution and, based on probabilistic strategies, suggests how to revise the norms (i.e., whether to alter, weaken or strengthen them) to continuously guarantee the achievement of the MAS objectives. We intend to consider a more complex case study and to extend our work with a number of additional aspects (e.g., the importance of the different norms, their mutual constraints, their conflicts, properties, synergies, etc.) that would offer a more realistic image of the complexity of enforcing multiple norms in a MAS. Incorporating a degree of norm violation, and altering the sanctions used to enforce the norms, is another interesting extension of this work. Empirical evaluation on multiple cases in different domains is also necessary to identify algorithms that perform well in different types of MAS. In the same direction, we intend to assess the robustness of the proposed approach to noise in data. Finally, in defining the norm revision operations, we treated any state and any proposition as equally important. In some cases, however, certain states or propositions may be more important than others and, in revising norms, one would expect to be able to consider such aspect. This is another future direction of our work. Similarly, the study here reported focused on a single MAS objective and on a boolean evaluation of such an objective. In future work, we also intend to extend the proposal to multiple labels, thereby supporting multiple objectives, and to more fine-grained evaluations (as opposed to the considered boolean evaluation) of the traces with respect to the objectives.

References

Ågotnes, T., van der Hoek, W., Rodríguez-Aguilar, J. A., Sierra, C., & Wooldridge, M. J. (2007). On the logic of normative systems. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI 2007*, pp. 1175–1180.

Airiau, S., Sen, S., & Villatoro, D. (2014). Emergence of conventions through social learning. *Autonomous Agents and Multi-Agent Systems*, *28*(5), 779–804.

Alechina, N., Bulling, N., Dastani, M., & Logan, B. (2015). Practical run-time norm enforcement with bounded lookahead. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015*, pp. 443–451. ACM.

Alechina, N., Dastani, M., & Logan, B. (2013). Reasoning about normative update. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence, IJCAI 2013*, pp. 20–26.

Alechina, N., Dastani, M., & Logan, B. (2014). Norm approximation for imperfect monitors. In *Proceedings of the 13th International conference on Autonomous Agents and Multi-Agent Systems, AAMAS 2014*, pp. 117–124.

Alechina, N., Logan, B., & Dastani, M. (2018). Modeling norm specification and verification in multiagent systems. *IfCoLog Journal of Logics and their Applications, FLAP*, *5*(2), 457–490.

Artikis, A., & Pitt, J. (2001). A formal model of open agent societies. In *Proceedings of the Fifth International Conference on Autonomous Agents, AGENTS 2001*, pp. 192–193.

Athakravi, D., Corapi, D., Russo, A., Vos, M. D., Padget, J. A., & Satoh, K. (2012). Handling change in normative specifications. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2012*, pp. 1369–1370.

Aucher, G., Grossi, D., Herzig, A., & Lorini, E. (2009). Dynamic context logic. In *Proceedings of the Second International Workshop on Logic, Rationality, and Interaction, LORI 2009*, Vol. 5834, pp. 15–26.

Bicchieri, C. (2005). *The grammar of society: The nature and dynamics of social norms*. Cambridge University Press.

Boella, G., & van der Torre, L. W. N. (2004). Regulative and constitutive norms in normative multiagent systems. In *Proceedings of the Ninth International Conference on Principles of Knowledge Representation and Reasoning, KR 2004*, pp. 255–266.

Broersen, J. M., Dastani, M., Hulstijn, J., Huang, Z., & van der Torre, L. W. N. (2001). The BOID architecture: conflicts between beliefs, obligations, intentions and desires. In André, E., Sen, S., Frasson, C., & Müller, J. P. (Eds.), *Proceedings of the Fifth International Conference on Autonomous Agents, AGENTS*, pp. 9–16.

Bulling, N., & Dastani, M. (2016). Norm-based mechanism design. *Artificial Intelligence*, *239*, 97–142.

Chopra, A., van der Torre, L., Verhagen, H., & Villata, S. (Eds.). (2018). *Handbook of Multiagent Systems*. College Publications, London.

Christelis, G., & Rovatsos, M. (2009). Automated norm synthesis in an agent-based planning environment. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2009*, pp. 161–168.

Cohen, B. H. (2008). *Explaining psychological statistics*. John Wiley & Sons.

Cohen, J. (1968). Weighted kappa: nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin*, *70*(4), 213.

Corapi, D., Russo, A., Vos, M. D., Padget, J. A., & Satoh, K. (2011). Normative design using inductive learning. *Theory and Practice of Logic Programming, TPLP*, *11*(4-5), 783–799.

Dastani, M., Grossi, D., Meyer, J. C., & Tinnemeier, N. A. M. (2009). Normative multi-agent programs and their logics. In *Normative Multi-Agent Systems, 15.03. - 20.03.2009*, Vol. 09121 of *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl.

Dell'Anna, D., Alechina, N., Dalpiaz, F., Dastani, M., & Logan, B. (2022a). Supplementary Material for "Data-Driven Revision of Conditional Norms in Multi-Agent Systems". Zenodo. https://doi.org/10.5281/zenodo.5907522.

Dell'Anna, D., Alechina, N., Logan, B., Löffler, M., Dalpiaz, F., & Dastani, M. (2022b). The complexity of norm synthesis and revision. In *Proceedings of the 15th International Workshop on Coordination, Organizations, Institutions, Norms, and Ethics for Governance of Multi-Agent Systems, COINE@AAMAS 2022*.

Dell'Anna, D., Dastani, M., & Dalpiaz, F. (2019). Runtime revision of norms and sanctions based on agent preferences. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2019*, pp. 1609–1617.

Dell'Anna, D., Dastani, M., & Dalpiaz, F. (2020). Runtime revision of sanctions in normative multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, *34*(2), 1–54.

Fitoussi, D., & Tennenholtz, M. (2000). Choosing social laws for multi-agent systems: Minimality and simplicity. *Artificial Intelligence*, *119*(1), 61–101.

Governatori, G., & Rotolo, A. (2010). Changing legal systems: legal abrogations and annulments in defeasible logic. *Logic Journal of the IGPL*, *18*(1), 157–194.

Jiang, J., Aldewereld, H., Dignum, V., & Tan, Y. (2012). Norm contextualization. In *Proceedings of the 14th International Workshop on Coordination, Organizations, Institutions, and Norms in Agent Systems, COIN 2012*, pp. 141–157.

Kafali, Ö., Ajmeri, N., & Singh, M. P. (2017). Kont: Computing tradeoffs in normative multiagent systems. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, pp. 3006–3012.

Kafali, Ö., Ajmeri, N., & Singh, M. P. (2020). DESEN: specification of sociotechnical systems via patterns of regulation and control. *ACM Transactions on Software Engineering and Methodology*, *29*(1), 7:1–7:50.

Katzouris, N., Artikis, A., & Paliouras, G. (2015). Incremental learning of event definitions with inductive logic programming. *Machine Learning*, *100*(2), 555–585.

Knobbout, M., & Dastani, M. (2012). Reasoning under compliance assumptions in normative multiagent systems. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2012*, pp. 331–340.

Knobbout, M., Dastani, M., & Meyer, J. C. (2016). A dynamic logic of norm change. In *Proceedings of the 22nd European Conference on Artificial Intelligence, ECAI 2016*, Vol. 285, pp. 886–894.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence, IJCAI 95*, pp. 1137–1145.

Kota, R., Gibbins, N., & Jennings, N. R. (2008). Decentralised structural adaptation in agent organisations. In *Proceedings of the First International Workshop on Organized Adaption in Multi-Agent Systems, OAMAS 2008*, Vol. 5368, pp. 54–71.

Krajzewicz, D., Erdmann, J., Behrisch, M., & Bieker, L. (2012). Recent development and applications of SUMO - Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements*, *5*(3&4), 128–138.

Lavrac, N., & Dzeroski, S. (1994). *Inductive logic programming - techniques and applications.* Ellis Horwood series in artificial intelligence. Ellis Horwood.

Lorenz, R., Senoner, J., Sihn, W., & Netland, T. (2021). Using process mining to improve productivity in make-to-stock manufacturing. *International Journal of Production Research*, 1–12.

Loreti, D., Chesani, F., Ciampolini, A., & Mello, P. (2020). Generating synthetic positive and negative business process traces through abduction. *Knowledge and Information Systems*, *62*(2), 813–839.

Mahmoud, M. A., Ahmad, M. S., Yusoff, M. Z. M., & Mostafa, S. A. (2018). A regulative norms mining algorithm for complex adaptive system. In *Proceedings of the Third International Conference on Soft Computing and Data Mining, SCDM 2018*, Vol. 700, pp. 213–224.

Miralles, J. C., López-Sánchez, M., Salamó, M., Avila, P., & Rodríguez-Aguilar, J. A. (2013). Robust regulation adaptation in multi-agent systems. *ACM Transactions on Autonomous and Adaptive Systems, TAAS*, *8*(3), 13:1–13:27.

Morales, J., López-Sánchez, M., Rodríguez-Aguilar, J. A., Wooldridge, M. J., & Vasconcelos, W. W. (2015). Synthesising liberal normative systems. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015*, pp. 433–441.

Morales, J., Wooldridge, M., Rodríguez-Aguilar, J. A., & López-Sánchez, M. (2017). Evolutionary synthesis of stable normative systems. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017*, pp. 1646–1648.

Muggleton, S. H., Lin, D., Pahlavi, N., & Tamaddoni-Nezhad, A. (2014). Meta-interpretive learning: application to grammatical inference. *Machine learning*, *94*(1), 25–49.

Perelli, G. (2019). Enforcing Equilibria in Multi-Agent Systems. In *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2019*, pp. 188–196.

Savarimuthu, B. T. R., & Cranefield, S. (2011). Norm creation, spreading and emergence: A survey of simulation models of norms in multi-agent systems. *Multiagent and Grid Systems*, *7*(1), 21–54.

Schapire, R. E., & Singer, Y. (2000). Boostexter: A boosting-based system for text categorization. *Machine learning*, *39*(2-3), 135–168.

Shoham, Y., & Tennenholtz, M. (1995). On social laws for artificial agent societies: Off-line design. *Artificial intelligence*, *73*(1-2), 231–252.

Sims, M., Corkill, D., & Lesser, V. (2008). Automated organization design for multi-agent systems. *Autonomous agents and multi-agent systems*, *16*(2), 151–185.

Sorower, M. S. (2010). A literature survey on algorithms for multi-label learning. *Oregon State University, Corvallis*, *18*, 1–25.

Testerink, B., Dastani, M., & Bulling, N. (2016). Distributed controllers for norm enforcement. In *Proceedings of the 22nd European Conference on Artificial Intelligence, ECAI 2016*, Vol. 285, pp. 751–759.

Tinnemeier, N. A. M., Dastani, M., Meyer, J. C., & van der Torre, L. W. N. (2009). Programming normative artifacts with declarative obligations and prohibitions. In *Proceedings of the 2009 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, IAT 2009*, pp. 145–152.

van der Aalst, W. M. P. (2016). *Process Mining - Data Science in Action, Second Edition*. Springer.

van der Hoek, W., Roberts, M., & Wooldridge, M. J. (2007). Social laws in alternating time: effectiveness, feasibility, and synthesis. *Synthese*, *156*(1), 1–19.

Vázquez-Salceda, J., Aldewereld, H., Grossi, D., & Dignum, F. (2008). From human regulations to regulated software agents' behavior. *Artificial Intelligence and Law*, *16*(1), 73–87.

Wooldridge, M., & van der Hoek, W. (2005). On obligations and normative ability: Towards a logical analysis of the social contract. *Journal of Applied Logic*, *3*(3), 396–420.

Wooldridge, M. J. (2009). *An Introduction to MultiAgent Systems (2. ed.)*. Wiley.