Online Automated Synthesis of Compact Normative Systems

JAVIER MORALES, Artificial Intelligence Research Institute (IIIA-CSIC), University of Barcelona, Spain
MAITE LÓPEZ-SÁNCHEZ, University of Barcelona, Spain
JUAN A. RODRIGUEZ-AGUILAR, Artificial Intelligence Research Institute (IIIA-CSIC), Bellaterra, Spain
WAMBERTO VASCONCELOS, University of Aberdeen, United Kingdom
MICHAEL WOOLDRIDGE, University of Oxford, United Kingdom

Most normative systems make use of explicit representations of norms (namely, obligations, prohibitions, and permissions) and associated mechanisms to support the self-regulation of open societies of self-interested and autonomous agents. A key problem in research on normative systems is that of how to synthesise effective and efficient norms. Manually designing norms is time consuming and error prone. An alternative is to automatically synthesise norms. However, norm synthesis is a computationally complex problem. We present a novel online norm synthesis mechanism, designed to synthesise compact normative systems. It yields normative systems composed of concise (simple) norms that effectively coordinate a multiagent system (MAS) without lapsing into overregulation. Our mechanism is based on a central authority that monitors a MAS, searching for undesired states. After detecting undesirable states, the central authority then synthesises norms aimed to avoid them in the future. We demonstrate the effectiveness of our approach through experimental results.

Categories and Subject Descriptors: I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms: Algorithms

Additional Key Words and Phrases: Normative systems, norm synthesis

ACM Reference Format:
Javier Morales, Maite López-Sánchez, Juan A. Rodriguez-Aguilar, Wamberto Vasconcelos, and Michael Wooldridge. 2015. Online automated synthesis of compact normative systems. ACM Trans. Autonom. Adapt. Syst. 10, 1, Article 2 (March 2015), 33 pages.
DOI: http://dx.doi.org/10.1145/2720024

1. INTRODUCTION

A *norm* is an established pattern of behaviour with which members of a society are expected to comply [Bicchieri 2006]. Typically, norms impose restrictions on the

This work was funded by AT (CONSOLIDER CSD2007-0022), EVE (TIN2009-14702-C02-01,TIN2009-14702-C02-02), COR (TIN2012-38876-C02-01/02), MECER (201250E053), and the Generalitat of Catalunya (2009-SGR-1434). Michael Wooldridge was supported by the ERC under advanced grant 291528 ("RACE").
Authors' addresses: J. Morales and J. A. Rodríguez-Aguilar, Artificial Intelligence Research Institute, Spanish Council of Scientific Research (IIIA-CSIC). Campus de la UAB, E-08193 Bellaterra, Catalonia (Spain); M. López-Sánchez, MAiA Department, University of Barcelona. Gran Via, 585 08007 Barcelona (Spain); W. Vasconcelos, Department of Computing Science, University of Aberdeen. Meston Building Meston Walk, Aberdeen, AB24 3UE, (United Kingdom); M. Wooldridge, Department of Computer Science, University of Oxford. Wolfson Building, Parks Rd, Oxford OX1 3QD (United Kingdom).

DOI: http://dx.doi.org/10.1145/2720024

behaviour of individuals. These behavioural restrictions play two important roles. On the one hand, they are of benefit to individuals because they simplify decision making by ruling out various courses of action, thus reducing the decision space of alternatives that need to be considered. On the other hand, at the social level, they provide means whereby agents in societies can *coordinate* their activities, using only local decision making [Binmore 2005].

Norms have been widely studied as a mechanism for coordinating multiagent systems (MAS) [Shoham and Leyton-Brown 2009; Dignum 1999; Boella et al. 2006]. Coordination in this sense is usually understood as achieving some system-level goal, such as ensuring that the system avoids certain undesirable states. However, the problem of actually *synthesising* norms that effectively coordinate a MAS is challenging. Since the seminal work of Shoham and Tennenholtz [1995], the norm synthesis problem (namely, creating a set of norms that ensures coordination is successful) has attracted considerable attention. Two approaches for norm synthesis have been considered in the literature: *offline* and *online*.

Offline approaches (e.g., Shoham and Tennenholtz [1995] and Fitoussi and Tennenholtz [1998]) are aimed at synthesising normative systems at design time. Unfortunately, the complexity of the offline norm synthesis problem is high (NP-hard) [Shoham and Tennenholtz 1995]. These complexity issues have prompted research into the problem of managing the size of the system state space [Christelis and Rovatsos 2009]. Unfortunately, even if we ignore the problem of computational complexity, computing norms offline is not appropriate if the state space of the system is not known in advance, or if it may change over time.

In contrast to offline approaches, online approaches are aimed at synthesising norms at runtime rather than design time. The key conceptual advantage of online approaches compared to offline approaches is that online approaches are not assumed to require complete knowledge of the system at design time.

Recently, *norm emergence* (or convention emergence) has become a popular technique for online norm synthesis (e.g., see Sen and Airiau [2007], Sen and Sen [2010], Griffiths and Luck [2010], Salazar et al. [2010], Villatoro et al. [2011], and Yu et al. [2013]). Norm emergence does not require any global state representation or centralised control and considers that agents collaboratively choose their own norms. Norm emergence therefore implies that agents are endowed with the computational capability to synthesise norms, and that they will choose to cooperate in the norm synthesis process. A norm is considered to have emerged when a significant number of agents in an agent society adhere to a common behaviour (i.e., they choose the same actions), which is not dictated by a central authority. Thus, a key issue in norm emergence is the design of emergence mechanisms that help agents agree online (converge) on some norm(s) [Kittock 1993; Walker and Wooldridge 1995]. Typically, state-of-the-art norm emergence mechanisms (1) require an initial set of predesigned, alternative norms; (2) are sensitive to such initial conditions; and (3) mainly converge to a unique norm instead of to a set of norms (with the exception of Sen and Sen [2010] and Salazar et al. [2010]).

A recent, alternative online approach is described in Morales et al. [2011c]. There, norms are synthesised by observing agent interactions without requiring their active participation in the synthesis process, unlike state-of-the-art norm emergence mechanisms. The approach in Morales et al. [2011c] proved to be capable of synthesising a set of norms (i.e., a normative system), instead of a single norm, from *scratch* (namely, without requiring any initial, alternative norms). Moreover, it proved that the norms it synthesises are both effective and necessary to achieve MAS coordination. On the one hand, the *effectiveness* of a norm determines to what extent it is capable of avoiding undesired MAS states. On the other hand, the *necessity* of a norm can be employed to determine whether it is really necessary to avoid undesired states, or those undesired states could be avoided without the existence of the norm.

Typically, norms are synthesised with the aim of developing a *stable* set of norms that will prevent undesired states of the MAS (i.e., conflicts). However, assuming that agents are autonomous and that they can choose whether to comply or not with norms, we cannot guarantee the prevention of undesired states. Therefore, here we assume that a set of norms is stable if (1) norms avoid undesired states as long as agents comply with them and (2) the set of norms remains unchanged for an extended period of time. Norm stability is essential to providing agents with a common framework for their interactions. However, in addition to simply avoiding undesired states effectively, we might also want to consider *compactness* as a criteria for the online synthesis of norms. Compactness requires that the norms synthesised are as *small* as possible.

The work in Morales et al. [2011c] (hereafter referred to as BASE) considers compactness when synthesising norms. However, as we will show in this article, it suffers from two major limitations. First, compactness is significantly jeopardised by the way synthesised norms are evaluated and generalised. On the one hand, the evaluation process is ill defined because it unnecessarily aggregates the effectiveness and necessity of norms. On the other hand, general norms are synthesised with very little evidence, leading to the possibility of overgeneralisation. Second, the approach of Morales et al. [2011c] lacks *stability*: a set of norms is stable if it avoids undesired system states and remains unchanged for an extended period of time.

Notice that norm stability is essential to providing agents with a common framework for their interactions. As we will show later, the synthesis mechanism in Morales et al. [2011c] is rather unstable, only being able to synthesise a normative system that avoids undesired states when the number of norm infringements in the system is very low. This means that this mechanism frequently changes the rules of the game (the norms) to the agents in an agent society.

Against this background, the main contribution of this article is a new, domain-independent norm synthesis mechanism called IRON (Intelligent Robust Online Norm synthesis machine) to synthesise norms that characterise necessary conditions for coordination while avoiding overregulation. Like BASE, IRON (available in the on-line appendix), assumes that (1) norms do not incorporate sanctions, and hence they can be regarded as soft constraints imposed to the agents, and (2) agents employ a stochastic norm infringement model, although rational norm infringements would guide towards useful normative systems in a more informed way. In this realm, we show that IRON significantly outperforms the approach described in Morales et al. [2011c] in terms of the following:

—*Stability*: Unlike the approach in Morales et al. [2011c], IRON is *highly stable*, capable of synthesising enduring normative systems that avoid undesired states despite a high number of infringements in the agent society. We explore the limits of IRON to empirically show that its synthesis mechanism manages to synthesise norms even when the probability of agents violating norms is high, namely when half of the decisions of each agent may result in norm infringements.
—*Compactness*: IRON manages to converge to normative systems that are between 30% and 70% more compact (have fewer norms) than those synthesised by the approach of Morales et al. [2011c].

These advantages stem from the core components of IRON's abstract architecture, namely (1) an evaluation method that evaluates norms in terms of different synthesis criteria, which allows IRON to be stable; (2) a generalisation operator that allows IRON to synthesise compact normative systems, only generalising norms when there is enough evidence; and (3) a specialisation operator that makes it possible to undo underperforming generalisations.

Notice that although IRON was originally introduced in Morales et al. [2013], we provide several extensions in this article. First, here we provide a comprehensive description of IRON's norm generalisation mechanism, which is a core process within its synthesis process. Second, we introduce metrics for compactness of normative systems and provide a thorough empirical evaluation of IRON's compactness. Third, we provide a detailed comparison of IRON versus the approach in Morales et al. [2013] in terms of stability and compactness.

The proposed approach can be then applied to a wide range of domains that can be regulated by means of norms. As an example, in a traffic scenario, IRON can be employed to synthesise traffic rules that prevent cars from colliding. Another example is the case of online communities, where individuals interact by uploading and sharing opinions about different contents. There, users can also complain about contents they find to be inappropriate. In Morales et al. [2014], IRON has been proven to be capable of synthesising norms that prevent the users of an online community from uploading inappropriate contents.

The remainder of this article is organised as follows. Section 2 introduces the basic framework within which we work and defines the norm synthesis problem that we tackle in this article. Section 3 summarises the approach of Morales et al. [2011c] to facilitate comparison with IRON, which is described in Section 4. Section 5 presents our empirical evaluation of IRON. Section 6 draws some conclusions and discusses the applicability and limitations of our approach. Finally, Section 7 describes some possible avenues for future research.

2. BASIC DEFINITIONS AND PROBLEM STATEMENT

In this section, we provide some basic definitions and use these to formally state the problem of norm synthesis.

2.1. Basic Definitions

We consider a system composed of a finite set of agents $Ag = \{ag_1, \ldots, ag_n\}$ with a shared finite set of actions $Ac = \{ac_1, \ldots, ac_m\}$ that these agents can perform. Let S be the set of all possible *states* of the system, and let $C \subseteq S$ be a set of undesired (conflicting) states. We do not require any specific semantics for the notions of state and undesired state: the interpretation will depend on the particular domain of interest. As an example, consider a traffic scenario in which agents correspond to computer-controlled cars. In this case, the set of undesired states would correspond to those states containing collisions, for example.

We will use a language $\mathcal{L}$ to describe the states of a MAS. This language, to be more formally defined later, is a logical language containing the standard classical connectives, and a notion of consequence defined for it via a relation $\models$. Given a state $s \in S$, we let $\nu(s)$ denote an expression in $\mathcal{L}$ that describes the state. For instance, if $\mathcal{L}$ is a predicate logic language with grounded terms and s stands for a state of a traffic junction, $\nu(s)$ would be composed of the predicates describing the position of each car in it.

We assume that each agent has its own local view of the state of the system of which it is part. For instance, a car located at a road junction will have its own local perception of the system, which corresponds to the perception of cars in its vicinity (i.e., the junction) without including those other cars farther away in the road. Thus, an *agent context* is an agent's internal representation of a system state (i.e., its beliefs). Agents express their contexts in terms of an *agent language* $\mathcal{L}_{Ag}$, which we will detail at a later point. However, we often find it convenient to assume that the language is that of a predicate logic with grounded terms. We denote the consequence relation for this logic as $\models$. Henceforth, given a state $s \in S$ and an agent $ag \in Ag$ that is part of s, we will refer to its context by means of function $c : Ag \times S \to \mathcal{L}_{Ag}$.

We now introduce our notion of norm, which establishes obligations, permissions, and/or prohibitions [Meyer and Wieringa 1993] to an individual agent whenever some preconditions are fulfilled. Considering that in this work we focus on synthesising norms that can be easily interpreted and fulfilled by agents, we employ the agent language to express norms. Therefore, norm preconditions are expressed as formulae of $\mathcal{L}_{Ag}$ and hence in terms of an agent's point of view.

Definition 1 (*Norm*). A *norm* is a pair $\langle \varphi, \theta(ac) \rangle$ where $\varphi \in \mathcal{L}_{Ag}$ stands for the precondition of the norm, $ac \in Ac$ is an action, and $\theta \in \{obl, perm, prh\}$ is a deontic operator: *obl* indicates an *obligation*, *perm* indicates a *permission*, and *prh* indicates a *prohibition*.

An agent $ag \in Ag$ evaluates whether a norm $n = \langle \varphi, \theta(ac) \rangle$ applies in a state s as follows. First, we say that the context of ag in s, $c(ag, s)$, satisfies the precondition of norm n if and only if $c(ag, s) \models \varphi$. In this case, norm n applies to agent ag and the deontic expression $\theta(ac)$ will hold for it. More concretely, we assume that the precondition φ is a set of first-order predicates $p(\tau_1, \ldots, \tau_n)$, with p being a predicate symbol and $\tau_1, \ldots, \tau_n$ being terms of $\mathcal{L}_{Ag}$ (the set of predicates represents their conjunction); $\theta(ac)$ is an atomic deontic formula. We represent a *normative system* Ω as a set of norms.

Notice that a normative system in our model essentially consists of a set of soft constraints on the behaviour of agents. It could be argued that this approach is more specific than is strictly necessary: an alternative approach would be for a norm designer simply to specify the undesirable system states (conflicts) that agents should avoid. The advantage of such an alternative approach is that it provides agents with more flexibility than IRON: it leaves an agent free to choose how to respect the norm. However, there are at least two difficulties with such an alternative approach. First, it presents agents with the problem of determining for themselves how to act in such a way as to respect the norm—that is, each agent should then solve a potentially complex norm compliance problem. Second, such an approach may cause problems if there are multiple possible norms that satisfy the designers intent, each involving multiple agents. In this case, communication and cooperation may be required to ensure that agents correctly coordinate on the selection of just one of the available set of norms. At the very least, this will impose an additional communication overhead on agents, and at worst, it may be impossible in time-constrained settings.

We now introduce a running example to be used throughout the remainder of the article. We consider a traffic junction scenario composed of two orthogonal roads. Agents are autonomous cars that enter the scenario and cross the junction to reach their destinations. A car perceives the scenario by means of its local context, which is expressed by means of three unary predicate symbols $\{left, front, right\}$ of $\mathcal{L}_{Ag}$, which are used to represent what occupies the three road positions the car perceives. Each predicate has a single term from $\{car, bike, private, ambulance, police, fire\text{-}brigade, emergency, nil\}$ of $\mathcal{L}_{Ag}$, representing different vehicle types, and the symbol "*nil*" standing for no vehicle. The actions available to agents are $Ac = \{Go, Stop\}$. In particular, the action *Go* means that the car moves forward in its target direction (in particular, it may include turns), and *Stop* means that the agent remains stopped in its position.

Using this traffic scenario, we can represent a norm that establishes a prohibition to go for an agent that observes a car to its left (hence giving way to it) as

$$n_0 : \langle \{left(car), front(nil), right(nil)\}, prh(Go) \rangle.$$

Where the precondition has a predicate $left(car)$ that is true if there is a car to the left of the agent evaluating the norm (the reference car), $front(nil)$ and $right(nil)$ are predicates that are true when, respectively, the front and right positions of the reference

car are empty. So, if agent ag's context is $\{left(car), front(nil), right(nil)\}$, then $c(ag) \models \varphi$ holds and $prh(Go)$ applies to the car.

Norm n_1 prohibits a car from going (hence giving way) if there is an ambulance to its left, a car to its front, and nothing to its right; it works in conjunction with two additional norms n_2 and n_3 to regulate priority of emergency vehicles:

$$\begin{aligned} n_1 &: \langle\{left(ambulance),\ front(car), right(nil)\},\ prh(Go)\rangle \\ n_2 &: \langle\{left(police),\ front(car), right(nil)\},\ prh(Go)\rangle \\ n_3 &: \langle\{left(fire\text{-}brigade), front(car), right(nil)\},\ prh(Go)\rangle. \end{aligned}$$

Let us suppose that $\{n_1, n_2, n_3\}$ comprise a normative system Ω. We observe that we can *generalise* these norms into a single norm that regulates when to give way to *emergency* vehicles coming from the left:

$$n_5 : \langle\{left(emergency), front(car), right(nil)\},\ prh(Go)\rangle.$$

As we explain next, this generalisation is possible because n_5 caters to the same situations (preconditions) of $\{n_1, n_2, n_3\}$ and establishes the same norm. Notice thus that whenever norms $n_1, n_2,$ or n_3 are applicable, their parent norm n_5 will be applicable as well. Let us now consider a norm similar to n_3, which prohibits a car from going if there is a fire brigade vehicle to its left, nothing to its right, and a bike in front:

$$n_4 : \langle\{left(fire\ brigade), front(bike), right(nil)\},\ prh(Go)\rangle.$$

Similarly to the generalisation of n_1, n_2, and n_3 into n_5, we may generalise n_3 and n_4 as a single norm establishing that agents should not go (and thus yield to a fire brigade) when there is a *private* vehicle in front (either a *car* or a *bike*):

$$n_6 : \langle\{left(fire\ brigade), front(private), right(nil)\},\ prh(Go))\rangle.$$

Next, we define how to establish such a relationship between norms. We use a subsumption relationship ($\sqsubseteq$) between the terms in $\mathcal{L}_{Ag}$. Thus, if $\tau, \tau' \in \mathcal{L}_{Ag}$ and $\tau' \sqsubseteq \tau$, we say that τ is more general than τ'. In our example, $police \sqsubseteq emergency$, as a police car "is a" specific kind of emergency vehicle.

Definition 2 (*Norm Generalisation*). We say that norm $n = \langle\varphi, \theta(ac)\rangle$ is more general than norm $n' = \langle\varphi', \theta(ac)\rangle$, denoted as $n' \subseteq n$, if and only if $|\varphi| = |\varphi'|$, and for each predicate $p(\tau'_0, \ldots, \tau'_n) \in \varphi'$, there is a predicate $p(\tau_0, \ldots, \tau_n) \in \varphi$ such that $\tau'_i \sqsubseteq \tau_i, 0 \leq i \leq n$.

In general, if n_j is *generalised* by n_i, then we also say that n_i is *specialised* by n_j. If there exists at least some $n_k \in \mathcal{N}$ such that $n_j \subseteq n_k \subseteq n_i$, we say that n_i is an *ancestor* of n_j, and otherwise n_i is a *father* of n_j. If n_j is not *generalised* by n_i, we denote it by $n_j \nsubseteq n_i$.

2.2. Research Problem

We evaluate norms and normative systems in terms of their effectiveness and necessity in achieving coordination. On the one hand, our system IRON measures the *cumulative* effectiveness of a norm from the outcomes of its *fulfilments*: the higher the ratio of *successful* fulfilments (fulfilments that did not lead to an undesired MAS state), the more effective the norm. On the other hand, it measures the necessity of a norm according to the following principle: the higher the ratio of *harmful infringements (infringements* leading to undesired states), the more necessary the norm.

In this article, we focus on a particular type of MAS, namely a normative multiagent system (NMAS). A NMAS is one whose agents have their actions regulated by some

normative system (set of norms) of which they are aware. Moreover, the system itself can assess whether and to whom norms in the normative system apply. For the sake of clarity, although NMAS is a concept used in the literature [Singh et al. 2013], the specific notion of NMAS that we use is as follows.

Definition 3 (*Normative MAS*). A normative multiagent system (NMAS) is a tuple $\langle Ag, Ac, \Omega, \mathcal{L}_{Ag}, S_0, S\rangle$, where (1) Ag is a set of agents; (2) Ac is a set of actions; (3) Ω is a normative system, whose norms are expressed in the agent language $\mathcal{L}_{Ag}$; (4) S_0 is the initial MAS state; and (5) S is a set of states.

Given a NMAS, our aim is to generate a normative system that avoids undesired states as long as agents comply with them while avoiding overregulation. With this aim, given a normative system, we must be able to measure (1) the effectiveness of its norms in preventing undesired states when agents comply with them and (2) whether its norms are necessary to prevent undesired states (i.e., if agents infringing norms lead to undesired states or not). Furthermore, since the state of a NMAS changes as agents interact, our goal is to find a *stable* normative system, namely a set of norms that remains unchanged and whose norms are sufficiently effective and necessary for a given period of time. With this aim, we make use of functions μ_{eff} and μ_{nec}, which are detailed in Section 4.3, that measure the effectiveness and necessity of a norm, respectively. We use these functions in tandem with (1) a time period $\mathbb{T}$ in which the norms of the normative system are required to be sufficiently effective and necessary to consider the normative system as stable, and (2) threshold values α_{eff}^{spec} and α_{nec}^{spec}, where $\alpha_{eff}^{spec}, \alpha_{nec}^{spec} \in [0, 1]$. These thresholds set satisfaction degrees for effectiveness and necessity. Thus, any norm to which effectiveness or necessity is under respective thresholds will be removed from the normative system. Next, we define the problem that we address in this article.

Definition 4 (*Norm Synthesis Problem*). Let $\langle Ag, Ac, \Omega, \mathcal{L}_{Ag}, S\rangle$ be a NMAS, $C \subseteq S$ a set of undesired states, μ_{eff}, μ_{nec} functions to assess the effectiveness and necessity of a norm, $\alpha_{eff}, \alpha_{nec}$ satisfaction degrees, ψ a function that returns the normative system at each given time t, t_0 the initial time step, and $T = [t_{begin}, \ldots, t_{end}]$ a time interval. The norm synthesis problem is that of finding a normative system $\bar{\Omega}$ such that for some $t_{begin} \geq t_0$, and for all t in T, the following conditions hold: (1) for each $n \in \psi(t)$, $\mu_{eff}(n, t, C) > \alpha_{eff}^{spec}$ and $\mu_{nec}(n, t, C) > \alpha_{nec}^{spec}$ and (2) $\psi(t) = \bar{\Omega}$.

Notice that solving this problem amounts to finding a stable normative system for a given period of time. Of course, it may be that several normative systems might be sufficiently effective and necessary during a period of time. To further distinguish between them, we use a measure of *compactness*: we prefer *smaller* normative systems over larger ones.

Definition 5 (*Compactness*). The compactness of a normative system Ω is the total number of predicates in its norms, namely $\sum_{n\in\Omega} |n|$, where $|n|$ stands for the number of predicates in a norm.

3. BACKGROUND

In this section we survey BASE, the norm synthesis approach described in Morales et al. [2011c], as our work builds on this framework. BASE is an iterative approach that continuously monitors a system, searching for undesired states. Whenever an undesired state is detected, a *norm generation* process is initiated, which results in the generation of a new norm aimed at avoiding that undesired state in the future. This generation process is based on an unsupervised approach of case-based reasoning (CBR).

ALGORITHM 1: BASE norm synthesis strategy

```
      Input: Ω, T_w, α_deact
      Output: Ω
      /* Norm generation                                          */
[1]   for conflict ∈ detectedConflicts do
[2]       n ← generateNorm(conflict);
[3]       Ω ← Ω ∪ {n};
      end
      /* Norm evaluation                                          */
[4]   F ← fulfilledNorms(Ω);
[5]   I ← infringedNorms(Ω);
[6]   for n ∈ (F, I) do
[7]       U(n, T_w) ← evaluate(n, F, I, T_w);
      end
      /* Norm refinement                                          */
[8]   for n ∈ (F, I) do
[9]       if U(n, T_w) < α_deact then
[10]          Ω ← deactivate(Ω, n)
          end
      end
[11]  return Ω
```

Classical CBR [Riesbeck and Schank 1989; Aamodt and Plaza 1994] is a supervised machine learning technique that solves new problems (i.e., cases) by obtaining similar ones from a knowledge base (i.e., a case base) and adapting their solutions under the supervision of an expert. The unsupervised CBR in Morales et al. [2011c] starts with an initially empty case base and does not require an expert to evaluate generated solutions. Instead, cases (and their solutions) are elicitedat runtime and evaluated in an unsupervised manner. Whenever BASE detects an undesired MAS state, it generates (and adds to the case base) a new case that describes the undesired state (at time t) and the previous state (at time $t-1$) that led to that undesired state.[1] Then, it generates a new solution that is aimed to avoid the transition to that undesired state in the future. To generate solutions, it randomly chooses one of the agents involved in the transition (i.e., in a traffic scenario, one of the agents that have collided). Then, it prohibits the action that the agent performed in the transition from the previous to the undesired state (i.e., from time $t-1$ to t). The generated solution is finally translated into a norm that agents can interpret and with which they can comply. Generated norms (and their corresponding solutions) are evaluated at runtime based on their outcomes in the MAS.

Next, a *norm evaluation* process is carried out, which computes the performance of those norms that have been fulfilled, as well as those that have been infringed, during the current step. Finally, a *norm refinement* process is carried out, which discards norms that underperform during a given period of time T_w.

Finally, as a result of norm generation, evaluation, and refinement, BASE outputs a normative system Ω. This normative system, which contains the currently prevailing norms, is then broadcast to the agents so that they become aware of it.

To provide the proper background to our approach, we present Algorithm 1 from Morales et al. [2011c]. It describes the BASE strategy for synthesising normative systems. In the norm generation phase, for each detected undesired MAS state (i.e., conflict), the function *generateNorm* (line 2) creates a new norm aimed at avoiding that

[1]This implies that BASE must continuously monitor the MAS by means of a window of two timesteps (i.e., the current MAS state at time t and the previous MAS state at time $t-1$).

particular state and adds the new norm to the normative system (Ω). Next, the norm evaluation process evaluates applicable norms—that is, those norms that have been fulfilled (F) and infringed (I) during the current step t (lines 4 and 5), in terms of their effectiveness and necessity to achieve MAS coordination. On the one hand, it measures the *cumulative* effectiveness of a norm from the outcomes of its *fulfilments*. Specifically, whenever agents fulfil a norm and it leads to a nonundesired state (i.e., a nonconflicting state), then the norm is considered as effective to avoid undesired states. By contrast, if agents comply with the norm and it leads to an undesired state, then the norm is considered as ineffective. Thus, the higher the ratio of *successful* fulfilments (fulfilments that did not lead to undesired MAS states), the more effective the norm. On the other hand, it measures the necessity of a norm based on the outcomes of its infringements. Whenever agents infringe on a norm and it leads to an undesired state, then the norm is considered as necessary. If agents infringe on the norm and it does not lead to an undesired state, then it is considered as unnecessary. Thus, the higher the ratio of *harmful infringements* (infringements leading to undesired MAS states), the more necessary the norm. The function *evaluate* (line 7) evaluates applicable norms by combining their effectiveness and necessity into a single utility measure u. Finally, the overall utility U of a norm is computed as the degree of positive evaluations (i.e., $u > 0$) over its total evaluations during a period of time T_w:

$$U(n, T_w) = \frac{\bar{u}_+(n, T_w)}{\bar{u}(n, T_w)}, \tag{1}$$

where $\bar{u}_+(n, T_w)$ stands for the average of the positive evaluations of n during the period of time T_w and $\bar{u}(n, T_w)$ stands for the average of all evaluations of n during period T_w. Finally, BASE carries out a norm refinement process, which deactivates those norms that underperformed during period T. Whenever the utility $U(n, T_w)$ of a norm is under a certain threshold α_{deact}, it is deactivated (line 10).

This approach can be seen to suffer from the following drawbacks:

—*Generalisation on the basis of scarce evidence*: Let us consider a traffic scenario and the following situation: a car perceives another car to its left, nothing in front, and nothing to its right. Let us suppose that we want to force a car to stop whenever it perceives this situation. An appropriate norm to generate would be *stop when there is a car to your left, nothing in front, and nothing to your right*. However, we observe that this norm synthesis instead generates a general norm, such as *stop when there is a car to your left*, disregarding the front and right positions. This norm should only be generated when there is evidence of *all* situations that it may represent. However, BASE generates this general norm on the basis of a single piece of evidence.

—*Ill-defined evaluation*: The norm evaluation process evaluates norms by aggregating their effectiveness and necessity into one unique value that represents their overall utility. This causes a coupling effect between effectiveness and necessity, and makes it impossible to evaluate whether norms are necessary *independently* of their effectiveness. For instance, a norm that is highly unnecessary but also highly effective may be evaluated as performing well since its resulting utility balances the differences between its effectiveness and necessity.

—*Lack of specialisation*: This problem arises when norms that do not perform well are simply removed from the normative system without taking into account the fact that underperforming general norms may be refined to eliminate just those cases that decrease their utility.

Despite the preceding drawbacks, one of the advantages of BASE is its low complexity, as shown by the following lemma.

LEMMA 3.1. *The complexity of the norm synthesis performed by the* BASE *algorithm employing a case base C containing* η_C *norms when detecting* κ *conflicts is* $O(\kappa \cdot \eta_C + 4 \cdot |Ag| \cdot |\Omega|)$.

PROOF. On the one hand, the cost of generating norms for all detected conflicts is $O(\kappa \cdot \eta_C)$, as it may require searching through the whole CBR base per conflict. Next, the cost of building the sets F and I of fulfilled and infringed norms is $O(2 \cdot |Ag| \cdot |\Omega|)$. This is because each norm in the normative system may be either fulfilled or infringed by each agent in the system. Since the total size of the sets F and I together is $|Ag| \cdot |\Omega|$, the cost of evaluating norms is $O(|Ag| \cdot |\Omega|)$. Therefore, altogether, the norm evaluation process is $O(3 \cdot |Ag| \cdot |\Omega|)$. Finally, the cost of the norm refinement process is $O(|Ag| \cdot |\Omega|)$, since, again, it requires a search over a list whose maximum size is $|Ag| \cdot |\Omega|$. Putting all together, the resulting worst-case complexity is $O(\kappa \cdot \eta_C + 4 \cdot |Ag| \cdot |\Omega|)$. □

4. IRON: A NORM SYNTHESIS MECHANISM

In this section, we introduce the IRON synthesis mechanism, a norm synthesis approach aimed at solving the norm synthesis problem formalised by Definition 4. IRON is intended to synthesise effective compact normative systems (i.e., normative systems that avoid undesired MAS states).

Although the basic operation of IRON follows the online approach of the BASE algorithm described in Section 3,IRON was designed to overcome the drawbacks of BASE. Specifically, given a NMAS, IRON operates by continuously iterating the following steps: (1) it monitors the operation of the system; (2) it decides upon the addition of brand new norms to the current normative system; (3) it evaluates whether the effectiveness and necessity of the normative system are within expected thresholds; (4) if required, it refines the normative system; and (5) it makes the normative system available to the agents. Therefore, notice that IRON continuously searches for a normative system *online* while agents in the system are operating.

IRON is based on five components, namely (1) a grammar for the synthesis of new norms, (2) the normative network (a data structure to represent normative systems and explored norms), (3) a method for the evaluation of norms and normative systems, (4) a set of operators that make it possible to transform one normative system into another, and (5) a strategy that specifies when to use such operators. Next, we describe each component in detail and explain IRON's architecture and computational model.

4.1. Grammar for Norm Synthesis

Our approach employs a grammar $\mathcal{G}$ to synthesise candidate norms of the form $\langle \varphi, \theta(ac) \rangle$ (cf. Definition 1). We have adapted our grammar from García-Camino et al. [2009], using as building blocks *atomic formulae* of the form $p^n(\tau_1, \ldots, \tau_n)$, with p being an n-ary predicate symbol and $\tau_1, \ldots, \tau_n$ terms of $\mathcal{L}_{Ag}$:

$$
\begin{array}{lll}
\text{Norm} & ::= & \langle \{\text{LHS}\}, \text{RHS} \rangle \\
\text{LHS} & ::= & \text{LHS, LHS} \mid \rho \\
\text{RHS} & ::= & \theta(Ac) \\
\theta & ::= & obl \mid perm \mid prh \\
Ac & ::= & ac_1 \mid ac_2 \mid \ldots \mid ac_n \\
\rho & ::= & p^n(\tau_1, \ldots, \tau_n) \\
\tau & ::= & c_1 | \ldots | c_n | x_1 | \ldots | x_m,
\end{array}
$$

where a_i and x_j are constants and variables, respectively. We represent as $\mathbf{N}$ the set of all norms that comply with the preceding grammar.

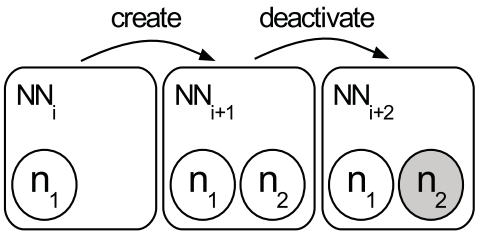

Fig. 1. Evolution of a normative network (and its normative system) along time: $\Omega_i = \{n_1\}$, $\Omega_{i+1} = \{n_1, n_2\}$, $\Omega_{i+2} = \{n_1\}$.

If we consider our traffic scenario further, the grammar used to synthesise the norm examples given in Section 2 is as follows:

$$\begin{array}{lll} Ac & ::= & Go \mid Stop \\ \rho & ::= & left(\tau) \mid front(\tau) \mid right(\tau) \\ \tau & ::= & car \mid bike \mid private \mid ambulance \mid police \mid fire\ brigade \mid emergency \mid nil. \end{array}$$

Terms τ in this grammar have the following subsumption relationships: $car \sqsubseteq private$, $bike \sqsubseteq private$, $ambulance \sqsubseteq emergency$, $police \sqsubseteq emergency$, $fire\ brigade \sqsubseteq emergency$.

4.2. A Representation of Normative Systems

IRON will continuously synthesise norms in search for a normative system; thus, it must be able to differentiate between the norms that are currently part of the normative system and those that are not (i.e., they have been explored but are not currently active). For this purpose, IRON employs a graph-based data structure to represent normative systems, which we call a *normative network*. A normative network is a graph whose nodes stand for norms and whose edges stand for relationships (more specifically, in our work, we concentrate on generalisation relationships) between norms. Norms in a network may be either *active* or *inactive*. We assume that a normative network represents a normative system as its active norms.

Figure 1 illustrates the evolution of a normative network (and its corresponding normative system) over time points t_i, t_{i+1}, t_{i+2}. At time t_i, the normative network NN_i has a single active norm n_1 (represented as a white circle) and $\Omega_i = \{n_1\}$. At time t_{i+1}, a new norm, n_2, is added to NN_i, yielding NN_{i+1} and $\Omega_{i+1} = \{n_1, n_2\}$. Finally, at time t_{i+2}, norm n_2 is deactivated (represented as a gray circle), giving rise to $NN_{i+2} = \{n_1, n_2\}$ and $\Omega_{i+2} = \{n_1\}$. Figure 1 also illustrates the way in which IRON performs the norm synthesis process. In general, the process will consist of continuously iterating over (applying changes to) the normative network accordingly until it finds a normative system that solves the norm synthesis problem.

We now offer a formal definition of the normative network employed by IRON.

Definition 6 (*Normative Network*). A normative network (NN) is a tuple $\langle \mathcal{N}, R_G, \Delta, \delta \rangle$, where (1) $\mathcal{N} \subseteq \mathbf{N}$ is a subset of our language of norms, (2) $R_G \subseteq \mathcal{N} \times \mathcal{N}$ is a *generalisation* relationship between norms, (3) $\Delta = \{active, inactive\}$ is the set of possible states of a norm, and (4) $\delta : \mathcal{N} \rightarrow \Delta$ is a function that returns the state of a norm $n \in \mathcal{N}$.

Since IRON considers that the current normative system is composed of the norms that are currently active in the normative network, we define $\Omega = \{n \mid n \in \mathcal{N}, \delta(n) = active\}$.

4.3. Evaluating Norms and Normative Systems

We detail how we evaluate norms and normative systems in terms of their effectiveness and necessity to achieve MAS coordination. IRON evaluates these two measures

individually, along the lines of BASE (described in Section 3). On the one hand, it measures the *cumulative* effectiveness of a norm from the outcomes of its *fulfilments*. The higher the ratio of *successful* fulfilments (fulfilments that did not lead to undesired states), the more effective the norm. On the other hand, it measures the necessity of a norm according from the outcomes of its infringements. The higher the ratio of *harmful infringements* (infringements leading to undesired states),the more necessary is the norm. We illustrate norm evaluation with the following example. In a traffic scenario, consider a norm like "give way to right." This is an effective norm, because agents will not collide with the car to their right as long as they comply with this norm. Thus, whenever agents fulfil this norm, it will lead to states that are not undesired (i.e., states without car collisions), which is why we call them successful fulfilments. Furthermore, this norm is also necessary to prevent cars from colliding with the cars to their right. Therefore, whenever agents infringe on the norm, it will lead to undesired states (i.e., states where cars have collided), which we refer to as a harmful infringement.

We compute the effectiveness $\mu_{eff}(n, t, C)$ of norm n up to time t as

$$\mu_{eff}(n, t, C) = (1 - \alpha) \times \mu_{eff}(n, t - 1, C) + \alpha \times r_{eff}(n, t, C), \tag{2}$$

where $\mu_{eff}(n, t - 1, C)$ is the effectiveness of n up to time $t - 1$ and $0 \leq \alpha \leq 1$ is a learning rate, and $r_{eff}(n, t, C)$ is a reward value based on the successful fulfilments of norm n at time t. In particular, at an initial time t_0, the effectiveness of a norm, namely $\mu_{eff}(n, t_0, C)$, is set with an initial constant value $k \in [0, 1]$. The reward of a fulfilled norm at time t is computed as

$$r_{eff}(n, t, C) = \frac{w_{SF} \times m_{SF}(n, t, C)}{w_{SF} \times m_{SF}(n, t, C) + w_{HF} \times m_{HF}(n, t, C)}, \tag{3}$$

where $m_{SF}(n, t, C)$ is the number of successful fulfilments of norm n at time t, namely the number of fulfilments of the norm that did not lead to undesired MAS states; $m_{HF}(n, t, C)$ is the number of harmful fulfilments of norm n at time t, namely the number of fulfilments that led to undesired states; and $w_{SF} > 0$ and $w_{HF} > 0$ weigh the importance of successful and harmful fulfilments of n, respectively. Notice that functions m_{HF} and m_{SF} receive as a parameter the set of undesired states C so as to assess if a norm fulfilment has led to an undesired state or not. As an example, we may decide that a harmful fulfilment (i.e., a norm fulfilment that leads to an undesired state) must be punished with a (negative) reward much higher than the (positive) reward that may obtain a successful fulfilment.

Therefore, we notice that our approach is akin to reinforcement learning [Sutton and Barto 1998] since both the effectiveness and necessity of a norm (μ_{eff}, μ_{nec}) are somehow *learned* by continuously aggregating rewards that have been computed from the outcomes of agents' norm compliance in the MAS.

Analogously, we compute the necessity of norm n up to time t as

$$\mu_{nec}(n, t, C) = (1 - \beta) \times \mu_{nec}(n, t - 1, C) + \beta \times r_{nec}(n, t, C), \tag{4}$$

where $\mu_{nec}(n, t - 1, C)$ is the necessity of n up to time $t - 1$, $0 \leq \beta \leq 1$ is a learning rate, and $r_{nec}(n, t, C)$ is a reward value based on the harmful infringements of norm n at time t. As well as with the effectiveness, the necessity of a norm at an initial time t_0 ($\mu_{nec}(n, t_0, C)$) is set to an initial constant value $k \in [0, 1]$. The reward of an infringed norm is computed as

$$r_{nec}(n, t, C) = \frac{w_{HI} \times m_{HI}(n, t, C)}{w_{HI} \times m_{HI}(n, t, C) + w_{SI} \times m_{SI}(n, t, C)}, \tag{5}$$

Table I. IRON Operators

Operator	Specification
$add(NN, n)$	$\mathcal{N}' \leftarrow \mathcal{N} \cup \{n\}$ $\delta'(n) \leftarrow active$ $NN' \leftarrow \langle \mathcal{N}', R_G, \Delta, \delta' \rangle$
$deactivate(NN, n)$	$\delta'(n) \leftarrow inactive$ $NN' \leftarrow \langle \mathcal{N}, R_G, \Delta, \delta' \rangle$
$generalise(NN, parent, children)$	$\mathcal{N}' \leftarrow \mathcal{N} \cup \{parent\}$ $R'_G \leftarrow R_G \cup \{(ch, parent) \mid ch \in children\}$ $\delta'(parent) \leftarrow active$ $\delta'(ch) \leftarrow inactive$ for all $ch \in children$ $NN' \leftarrow \langle \mathcal{N}', R'_G, \Delta, \delta' \rangle$

$specialise(NN, parent, children)$	$\delta'(parent) \leftarrow inactive$
	for all $child \in children$
	  if $(child, parent) \in R_G$
	  $\delta'(child) \leftarrow active$
	$NN' \leftarrow \langle \mathcal{N}, R_G, \Delta, \delta' \rangle$

where $m_{HI}(n, t, C)$ is the number of harmful infringements of norm n, namely infringements that led to undesired states; $m_{SI}(n, t, C)$ is the number of successful infringements of norm n, namely those infringements that did not lead to undesired states; and $w_{HI} > 0$ and $w_{SI} > 0$ weigh the importance of harmful and successful infringements of n, respectively. Moreover, IRON computes the effectiveness and necessity ranges of each norm during a time period T_w (i.e., a time window to compute the performance ranges of each norm). These ranges will be essential (as we show next) when performing generalisations and specialisations. We assess the effectiveness ($\breve{E}$) and necessity ($\breve{N}$) ranges for each norm as follows:

$$\breve{E} = [\bar{\mu}_{eff}(n, T_w, C) - \hat{\mu}_{eff}(n, T_w, C), \bar{\mu}_{eff}(n, T_w, C) + \hat{\mu}_{eff}(n, T_w, C)] \tag{6}$$

$$\breve{N} = [\bar{\mu}_{nec}(n, T_w, C) - \hat{\mu}_{nec}(n, T_w, C), \bar{\mu}_{nec}(n, T_w, C) + \hat{\mu}_{nec}(n, T_w, C)], \tag{7}$$

where $\bar{\mu}_{eff}(n, T_w)$ and $\hat{\mu}_{eff}(n, T_w)$ stand for the average and deviation of the effectiveness of n within T_w, respectively, and $\bar{\mu}_{nec}(n, T_w)$ and $\hat{\mu}_{nec}(n, T_w)$ stand for the average and deviation of the necessity of n within T_w, respectively.

4.4. Operators for Normative Networks

IRON will search for a normative system that solves the norm synthesis problem by transforming an initial normative network over time, hence moving from one normative system to another. With this aim, our norm synthesis mechanism implements a collection of normative network operators. Each operator transforms IRON's normative network $\langle \mathcal{N}, \mathcal{R}_G, \Delta, \delta \rangle$ into another one $\langle \mathcal{N}', \mathcal{R}'_G, \Delta, \delta' \rangle$. More specifically, IRON implements the following operators:

—The *addition* of a new norm into the normative system. As Table I formally specifies, whenever a new norm is synthesised, the `add` operator extends IRON's normative network with norm n ($\mathcal{N}' = \mathcal{N} \cup \{n\}$) and sets its state to active ($\delta'(n) = active$).
—The *deactivation* of a norm in the normative system. The implementation of this `deactivate` operator sets the state of a given norm to *inactive*. Hence, although the norm remains in the normative network, it is no longer part of the normative system.
—The *generalisation* of a set of norms in the normative system into a more general norm (e.g., considering the example in Section 2, generalising n_1, n_2, and n_3 into n_5). As Table I shows, this `generalise` operator generalises a set of norms (*children*) into a more general norm (*parent*) by (1) adding the parent norm to the network,

(2) establishing new generalisation relations (R_G) between each generalised (child) norm and the parent norm in the normative network, and (3) setting the state of the parent to active and that of the children to inactive. As a result, the child norms will no longer belong to the normative system, whereas the parent norm will.

—The *specialisation* of a norm in the normative system into more specific norms. This operation reverses the result of a generalisation (e.g., n_5 can be specialised into n_1, n_2, and n_3). Specifically, this `specialise` operator undoes the result of a generalisation by setting to inactive the state of the parent (more general) norm and setting to active the state of its children. Thus, thereafter, all child norms become candidates to belong to the normative system, whereas the parent norm does not any longer.

4.5. A Strategy to Synthesise Normative Systems

Algorithm 2 describes in outline IRON's overall norm synthesis strategy. Since IRON is an online mechanism, at every tick it runs its strategy to perform the same three main tasks of the BASE algorithm (Algorithm 1 in Section 3), namely (1) norm generation, (2) evaluation of the current normative system, and (3) refinement of the normative system. However, IRON implements them differently; thus, for example, refinement is done by means of specialisations, generalisations, and deactivation of norms. Once the strategy finishes, it returns the normative system represented by the normative network.

ALGORITHM 2: IRON strategy

Input: $\langle s_{t-1}, s_t \rangle, NN, \mathcal{G}, f_{apply}, f_{conflict}, \mu_{eff}, \mu_{nec}, \Theta, T_w$
Output: Ω
[1] $conflictDescription \leftarrow conflictDetection(s_{t-1}, s_t, f_{conflict})$;
[2] $NN \leftarrow normGeneration(NN, conflictDescription, \mathcal{G})$;
[3] $P \leftarrow normEvaluation(NN, \langle s_{t-1}, s_t \rangle, f_{apply}, f_{conflict}, \mu_{eff}, \mu_{nec}, T_w)$;
[4] $NN \leftarrow normRefinement(NN, P, \mathcal{G}, \Theta)$;
[5] $\Omega \leftarrow \{n \in NN | \delta'(n) = active\}$;
[6] **return** Ω

Algorithm 2 specifies IRON's general strategy (Π). It is domain independent and takes as input (1) a pair $\langle s_{t-1}, s_t \rangle$ containing descriptions of the system state at time $t-1$ and time t, respectively—this pair stands for a transition between the system state at consecutive times, and in fact, the differences between s_{t-1} and s_t reflect the local changes that occurred when the system evolved from $t-1$ to t; (2) a normative network NN, which includes the current normative system Ω; (3) a grammar $\mathcal{G}$, including the subsumption relationships between its terms; (4) a function f_{apply} to check norm applicability in the current system state s; (5) a function $f_{conflict}$ to detect if a given system state s is undesired; (6) two evaluation functions μ_{eff}, μ_{nec} to assess the effectiveness and necessity of norms in NN; (7) Θ, a set of satisfaction degree thresholds described later (see Equations (8), (9), (10), and (11)) ($\Theta = \{\alpha_{eff}^{gen}, \alpha_{nec}^{gen}, \alpha_{eff}^{esp}, \alpha_{nec}^{esp}\}$); and (8) a time period T_w.

The strategy is online and conflict driven (as it is aimed to avoid undesired, conflicting states), and thus at every tick, it starts by searching for conflicts in the current system state. Function *conflictDetection* (line 1) uses the domain-dependent function $f_{conflict}$ to assess if the current system state s_t is undesired (i.e., it detects undesired states $C \subset S$). In case $s_t \in C$, it returns a $conflictDescription$ that incorporates descriptions s_{t-1} and s_t, together with the identifiers of those agents whose actions lead to the undesired state s_t (i.e., in a traffic scenario, those cars that went forward before colliding). Next, the *normGeneration* function (line 2) synthesises a norm to avoid the transition from

s_{t-1} to s_t (although disregarding the generation of general norms) to avoid it in the future. Subsequently, *normEvaluation* in line 3 evaluates norms in terms of their effectiveness and necessity as discussed in Section 4.3. Finally, the norm refinement function in line 4 generalises and/or specialises norms according to their effectiveness and necessity ranges during the time period T_w, as discussed in Sections 4.6 and 4.7. The algorithm outputs a normative system (line 6) for the agents in the domain that IRON aims to regulate.

The main functions in IRON's strategy (i.e., in Algorithm 2) are specified in Algorithms 3, 4, and 5. As mentioned previously, our proposal is to monitor the evolution of the NMAS at regular time intervals (i.e., ticks) and apply operators under certain conditions. Next, we present how previously defined operators are invoked in the three main functions, which are specified as follows.

ALGORITHM 3: *normGeneration*

Input: NN, $conflictDescription$, $\mathcal{G}$
Output: NN
[1] **if** *isEmpty(conflictDescription)* **then**
[2] **return** NN
end
[3] $n \leftarrow generateNorm(\mathcal{G}, conflictDescription)$;
[4] $NN \leftarrow add(NN, n)$;
[5] $potential[n] \leftarrow potentialGeneralisations(n, \mathcal{G})$;
[6] **return** NN

ALGORITHM 4: *normEvaluation*

Input: NN, $\langle s_{t-1}, s_t \rangle$, f_{apply}, $f_{conflict}$, μ_{eff}, μ_{nec}, T_w
Output: P
[1] $applicableNorms \leftarrow normApplicability(NN, \langle s_{t-1}, s_t \rangle, f_{apply})$;
[2] $(F, I) \leftarrow normCompliance(applicableNorms, f_{conflict})$;
[3] **for** $n \in norms(F, I)$ **do**
[4] $U[n] \leftarrow updateUtilities(n, F, I, \mu_{eff}, \mu_{nec})$;
[5] $(\check{E}[n], \check{N}[n]) \leftarrow updatePerformanceRanges(n, U[n], T_w)$;
[6] $P[n] \leftarrow (\check{E}[n], \check{N}[n])$
end
[7] **return** P

ALGORITHM 5: *normRefinement*

Input: $NN, P, \mathcal{G}, \Theta$
Output: NN
[1] **for** $n \in norms(F, I)$ **do**
[2] **if** $underperforms(n, \check{E}, \check{N}, \alpha_{eff}^{spec}, \alpha_{nec}^{spec})$ **then**
[3] $NN \leftarrow specialiseDown(NN, n, \check{E}, \check{N}, \alpha_{eff}^{spec}, \alpha_{nec}^{spec})$;
else
[4] $NN \leftarrow generaliseUp(NN, n, \check{E}, \check{N}, \alpha_{eff}^{gen}, \alpha_{nec}^{gen})$;
end
[5] **return** NN
end

4.5.1. Generation of New Norms. As mentioned previously, IRON starts by detecting if the current system state s_t is undesired or not. Function *conflictDetection* (line 2 in Algorithm 2) uses function $f_{conflict}$ to assess if s_t belongs to the set of undesired states C, $s_t \in C$. Afterwards, if s_t is an undesired state, the norm generation function (line 2 in Algorithm 2, and Algorithm 3) synthesises a new norm to avoid the transition from s_{t-1} to s_t in the future. Specifically, it first invokes function *generateNorm* (line 3 in Algorithm 3), which employs the unsupervised CBR mechanism and the grammar $\mathcal{G}$ together with subsumption relationships of its terms (see Section 4.1) to generate a new norm aimed to avoid the transition from s_{-1} to s_t. Second, as line 4 in Algorithm 3 indicates, the newly created norm n is added to the normative network by invoking the add operator. Third, the norm generation function ends by creating all potential generalisations (see Algorithm 6 in Section 4.6) for each newly created norm n (line 5).

Basically, the implementation of this *generateNorm* function is the same as for the BASE approach described in Section 3 [Morales et al. 2011c]. The only difference is that our synthesis process does not generate general norms, as the generalisation process runs separately from norm generation. *generateNorm* is based on an unsupervised CBR approach. Hence, this approach is based on the following principle: if we can prevent a conflict in a given situation by enacting a norm, it is likely that we can prevent a conflict in a similar situation by means of a similar norm. Initially, when no other knowledge is available, norms are created based on the following heuristic: the transition to an undesired state is caused by the actions of the agents involved in that undesired state. Therefore, our *generateNorm* chooses one of the agents involved in an undesired state s_t and gathers its context and performed action in s_{t-1}. Then, the norm is synthesised so that this action is prohibited for any agent encountering this same context. This is done with the aim of avoiding the previously resulting conflict in future situations. Obviously, there is no certainty about the validity of this norm, and this is the reason IRON needs to continuously evaluate it in the NMAS.

Our norm synthesis uses ground atomic formulae from the states and focusses on a specific agent to figure out its context φ (what it was able to perceive, given its position in the grid) and the action ac it carried out (which possibly led to a conflicting state). The synthesised norm $\langle \varphi, prh(ac) \rangle$ is thus made up of fully instantiated, nonnegated atomic formulae, establishing the specific context of one agent and its action that led to a conflicting state. This process differs from induction of logical theories (e.g., inductive logic programming [Muggleton and Raedt 1994; Nienhuys-Cheng and Wolf 1997]) in two important ways: (1) the synthesised norm is specific with fully ground predicates (so we do not need to perform inverse resolution to synthesise norms with variables), and (2) negated atoms do not play any role, as these are not represented in states.

Note that implicit within our model is a reduction of deontic operators to prohibitions that forbid actions in certain conditions. It is natural to ask whether this represents a problematic restriction. However, notice that obligations and permissions are frequently and naturally interpreted as dual notions: an action is obligatory if it is not permitted to refrain from performing the action, and an action is permissible if it is not obligatory to refrain from performing the action. In this case, obligations can be reduced to prohibitions: to make an action obligatory, prohibit everything else. With this in mind, our approach of focussing on prohibitions can be seen also encompassing obligations. Of course, richer deontic operators (e.g., conditional obligations) have also been considered in the literature, and we do not claim that our approach encompasses a full range of these. But with respect to core operators, our approach is sufficient.

Moreover, our approach can also handle permissions. We observe that the interpretation of permissions can vary. For instance, we may have normative systems whereby all actions are prohibited unless they are explicitly permitted. Alternatively, we may

have systems whereby all actions are permitted unless explicitly prohibited, and permissions are ways of encouraging certain behaviours. When dealing with obligations and permissions, the undesired states are *idealised* situations that did not occur, and the context of an agent is used to establish an obligation or permission on a missing action (which would have reached the idealised situation).

4.5.2. Norm Evaluation. As mentioned previously, the strategy updates the effectiveness and necessity of norms by evaluating them individually. In general, IRON evaluates a norm depending on the outcome (conflicts) to which either its fulfilment or infringement has led. Therefore, norm evaluation will solely consider the applicable norms that have been either fulfilled or infringed by the agents in the NMAS in the transition from two consecutive states in time.

The norm evaluation function is specified by Algorithm 4. Function *normApplicability* (line 1) uses function f_{apply} to assess the norms in the normative network (*NN*) that were applicable at tick $t-1$. Thus, this function assesses the norms that are applicable at state s_{t-1}. Next, function *normCompliance* (line 2) partitions the selected applicable norms into fulfilled and infringed norms (F and I). Moreover, it uses a conflict-detection function ($f_{conflict}$) to determine which norms led to undesired states. As a result, we obtain a partition of applicable norms into four multisets (sets that allow duplicate values): (1) fulfilled norms that led to undesired MAS states (F_C), (2) fulfilled norms that did not lead to undesired MAS states ($F_{\bar{C}}$), (3) infringed norms that led to undesired MAS states (I_C), and (4) infringed norms that did not lead to undesired MAS states ($I_{\bar{C}}$). In the algorithm, $F = (F_C, F_{\bar{C}})$ and $I = (I_C, I_{\bar{C}})$. These sets are the ones used by function *updateUtilities* (line 4) to compute the effectiveness and necessity of each norm at time t. In fact, this function implements Equations (2) through (5) given in Section 4.3. Next, in line 5, function *updatePerformanceRanges* computes its effectiveness and necessity ranges during a period of time T_w (see Equations (6) and (7) in Section 4.3).

4.5.3. Normative System Refinement. The final function in our strategy is the *normative system refinement*, which yields a new normative system by transforming the normative network via specialisations and generalisations. Specifically, it specialises any underperforming norms (lines 2 and 3 in Algorithm 5) while it tries to generalise those norms that performed well (line 4). With this aim, the strategy keeps track of effectiveness and necessity ranges of the norms in the normative network during a period of time T_w. Then, the refinement task amounts to implementing the following rules:

—A norm is *specialised* (or *deactivated* if it has no children in the normative network) provided that either its effectiveness *or* necessity have not been good enough during T_w. This occurs when the effectiveness *or* necessity of some of its children have not been good enough either.
—A set of norms are *generalised* provided that (1) they all relate to the very same norm (parent) in the normative network, (2) they are all the possible child norms of the parent norm, and (3) their effectiveness *and* necessities have all been sufficiently high during T_w.

The following sections describe both processes in more detail.

4.6. Norm Generalisation

Norm generalisation starts whenever IRON detects some norm that has *performed well* during a period of time T_w. We say that a norm n has performed well during a period T_w if the lower bounds of its effectiveness *and* necessity ranges are above some satisfaction (generalisation) thresholds. This amounts to satisfying the following generalisation

Table II. $\check{E}$ and $\check{N}$ Ranges for Norms in Figure 2

	$\check{E}$	$\check{N}$
n_1	[0.6, 0.7]	[0.5, 0.6]
n_2	[0.7, 0.8]	[0.6, 0.7]
n_3	[0.8, 0.9]	[0.5, 0.7]
n_4	[0.7, 0.9]	[0.6, 0.8]
n_5	[0.6, 0.9]	[0.5, 0.8]
n_6	[0.6, 0.8]	[0.6, 0.9]

conditions:

$$\check{E}_{min}(n, T_w, C) > \alpha_{eff}^{gen} \tag{8}$$

$$\check{N}_{min}(n, T_w, C) > \alpha_{nec}^{gen}, \tag{9}$$

where $\check{E}_{min}(n, T_w, C)$ and $\check{N}_{min}(n, T_w, C)$ are the lower bounds of $\check{E}$ and $\check{N}$ that are described in Equations (6) and (7), and $\alpha_{eff}^{gen} \in [0, 1]$ and $\alpha_{nec}^{gen} \in [0, 1]$ are the generalisation thresholds for effectiveness and necessity.

Given a generalisable norm, the generalisation process is based on building all *potential* generalisations for the norm to subsequently analyse whether each one of them can be enacted or not. If a potential generalisation is enacted, it transforms the current normative system into another one. Otherwise, the normative system remains unchanged. Next, we illustrate (1) how to build potential generalisations and (2) how to enact potential generalisations. We illustrate our approach with norms n_1–n_6 from Section 2, their effectiveness and necessity ranges listed in Table II, and $\alpha_{eff}^{gen} = 0.5$, $\alpha_{nec}^{gen} = 0.4$ as generalisation thresholds.

4.6.1. How to Build Potential Generalisations. We now consider norm n_3 described in Section 2. The first step in building a potential generalisation is to find a more general parent norm. With this aim, our generalisation process employs a grammar $\mathcal{G}$ together with subsumption relationships of its terms (see Section 4.1) to build predicates of the language $\mathcal{L}_{Ag}$. In our example, with the aid of the grammar, n_5 would be built, which is more general than n_3 and thus can be a parent of a potential generalisation. The second step is to find all children of the parent. In our example, the children of n_5 (the siblings of n_3) are n_1 and n_2. The triple $\langle n_3, n_5, \{n_1, n_2\}\rangle$ records a potential generalisation—that is, $\langle n, n', S\rangle$, where n is a generalisable norm, n' is a parent norm of n, and S is a possibly empty set of any other norms that n' generalises (disregarding n). The generalisation process now continues to build all potential generalisations. In our example, the triple $\langle n_3, n_6, \{n_4\}\rangle$ would also be built as a potential generalisation of n_3.

4.6.2. How to Enact Potential Generalisations. A potential generalisation $\langle n, n', S\rangle$ can be enacted if and only if all siblings of n (the norms in S) belong to the normative network (i.e., they have been created previously by the norm generation process) and satisfy all generalisation conditions of Equations (8) and (9). If these conditions hold, the potential generalisation is enacted to transform the normative system. As an example, consider the two potential generalisations $\langle n_3, n_5, \{n_1, n_2\}\rangle$ and $\langle n_3, n_6, \{n_4\}\rangle$ and the normative network from Figure 2(a) showing $n_1, n_2, n_3,$ and n_4 as active norms (i.e., we have a normative system $\Omega_1 = \{n_1, n_2, n_3, n_4\}$). From Table II, we observe that norms $n_1, n_2,$ and n_4 fulfil the generalisation conditions in Equations (8) and (9), and so does n_3. Thus, the first potential generalisation can be enacted as follows: (1) norm n_5 is created, added to the normative network, and activated; (2) $n_1, n_2,$ and n_3 are deactivated in the normative network; and (3) generalisation relationships are established between $n_1, n_2, n_3,$

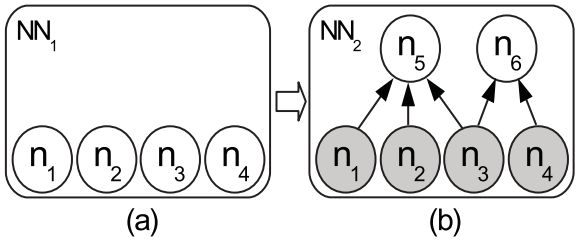

Fig. 2. Generalisation of n_1, n_2, n_3, n_4: norms n_5 and n_6 are active, whereas the rest are inactive, so the normative system consists of $\Omega = \{n_5, n_6\}$.

and n_5. This amounts to applying the `generalise` $(NN, n_5, \{n_1, n_2, n_3\})$ operator in Section 4.4. Analogously, we can enact the second potential generalisation by invoking `generalise` $(NN, n_6, \{n_4, n_3\})$. Figure 2(b) shows the resulting normative network after both norm generalisations. It contains two active norms (n_5, n_6) and four inactive norms (n_1, n_2, n_3, and n_4)—that is, we have the normative system $\Omega_2 = \{n_5, n_6\}$. These two norm generalisations reduce the size of the normative system from 4 norms to 2 and the total number of predicates from 12 to 6 (improving its compactness).

ALGORITHM 6: *generaliseUp*

Input: $n, \check{E}, \check{N}, NN, \alpha_{eff}^{gen}, \alpha_{nec}^{gen}$
Output: NN
[1] **for** $\langle n, n', S\rangle \in potential[n]$ **do**
[2] **if** $areGeneralisable(S, \check{E}, \check{N}, \alpha_{eff}^{gen}, \alpha_{nec}^{gen})$ **then**
[3] $NN \leftarrow generalise(NN, n', S \cup \{n\})$
end
end
[4] **return** NN

4.6.3. Generalisation Algorithm. Norm generalisations are performed by the function *generaliseUp* in Algorithm 6. Its inputs are a norm to generalise (n) along with its effectiveness and necessity ranges ($\check{E}, \check{N}$), a normative network (NN), and the thresholds to verify the generalisation conditions. For each potential generalisation of norm n (line 1), function *areGeneralisable* (line 2) verifies whether all norms in S (i.e., the children of the generalisation) satisfy the generalisation conditions. If so, the operator `generalise`$(NN, n', S \cup \{n\})$ (line 3) adds the parent norm to the normative network NN and activates it, deactivates the child norms, and establishes generalisation relationships between child norms and parent norm. Additionally, the auxiliary function *potentialGeneralisations* can be employed to generate all potential generalisations for a given norm n (indeed, as line 4 in Algorithm 3 shows, our strategy invokes this function when creating the norm). First, it takes the precondition φ of a norm n (line 1) and employs function *general* (line 2) to obtain its parent preconditions Φ by using grammar $\mathcal{G}$ together with subsumption relationships of the terms of $\mathcal{L}_{Ag}$. Second, in lines 3 and 4, for each parent precondition $\varphi' \in \Phi$ it builds n', a parent norm, based on the general precondition φ' and the same consequent $\theta(ac)$ from n (i.e., $n' = \langle \varphi', \theta(ac)\rangle$). Next, function *generateChildren*$(n', \mathcal{G}, n)$ (line 5) computes (again using grammar $\mathcal{G}$ and subsumption term relationships) child norms of n' disregarding n. Finally, in line 6, it builds a new potential generalisation $\langle n, n', S\rangle$ that is added to the potential generalisations of norm n.

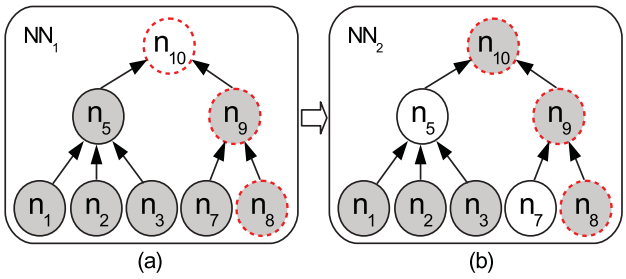

Fig. 3. Specialisation of norm n_{10} into norms n_5 and n_7. (a) $\Omega_1 = \{n_{10}\}$. (b) $\Omega_2 = \{n_5, n_7\}$.

ALGORITHM 7: *potentialGeneralisations*

Input: $n, \mathcal{G}$
Output: $potential[n]$
[1] **let** $n = \langle \varphi, \theta(ac) \rangle$;
[2] $\Phi \leftarrow general(\varphi, \mathcal{G})$;
[3] **for** $\varphi' \in \Phi$ **do**
[4] $n' \leftarrow \langle \varphi', \theta(ac) \rangle$;
[5] $S \leftarrow generateChildren(n', \mathcal{G}, n)$;
[6] $potential[n] \leftarrow potential[n] \cup \{\langle n, n', S \rangle\}$;
end
[7] **return** $potential[n]$

4.7. Norm Specialisation

Norm specialisations make it possible to *refine* generalisations of norms that do not perform well during a period of time. Take the normative network depicted in Figure 3(a) with a single active norm, $\Omega_1 = \{n_{10}\}$. The network contains norms n_1, n_2, n_3, and n_5 (defined in Section 2), together with

$$\begin{aligned}
n_7 &: \langle \{left(car), \quad front(car), right(nil)\}, prh(Go) \rangle \\
n_8 &: \langle \{left(bike), \quad front(car), right(nil)\}, prh(Go) \rangle \\
n_9 &: \langle \{left(private), front(car), right(nil)\}, prh(Go) \rangle \\
n_{10} &: \langle \{left(vehicle), front(car), right(nil)\}, prh(Go) \rangle.
\end{aligned}$$

Norms n_7 and n_8 prohibit a car from proceeding whenever there is a car or a bike to its left, a car in front, and nothing to its right. Norm n_9 is a generalisation of norms n_7 and n_8 to give way to private vehicles on the left, whereas norm n_{10} is a generalisation of all norms from n_1 to n_9 to give way to any vehicle on the left. Recall from Section 4.1 that IRON uses a grammar to construct norms. Each norm contains a precondition with three predicates $left$, $front$, $right$, and a postcondition with a prohibition to perform action Go. Notice that all norms from n_1 to n_{10} are constructed by means of this grammar. In Figure 3(a), norm n_{10} is the only active norm, and we assume that all norms below n_{10} have been generalised previously to n_{10}.

Norm specialisation starts out whenever IRON detects that some norm has underperformed during a period of time T_w. We say that a norm n has underperformed during a period T_w if the higher bound of its effectiveness range *or* the higher bound of its necessity range are below the satisfaction (specialisation) thresholds provided in the strategy (see Section 4.5). This amounts to satisfying either (or both) of the two

following conditions:[2]

$$\check{E}_{max}(n, T_w, C) < \alpha_{eff}^{spec} \tag{10}$$

$$\check{N}_{max}(n, T_w, C) < \alpha_{nec}^{spec}, \tag{11}$$

where $\check{E}_{max}(n, T_w, C)$ and $\check{N}_{max}(n, T_w, C)$ are the higher bounds of $\check{E}$ and $\check{N}$ that are described in Equations (6) and (7). Let us suppose that norms $n_1, n_2, n_3, n_5,$ and n_7 have performed well during a period T_w but norms $n_8, n_9,$ and n_{10} have not. This is a reasonable assumption. Since n_9 and n_{10} are more general than n_8, they also apply in the same situations as n_8. This means that the poor performance of n_8 affects n_9, whose performance in turn affects n_{10}. In general, the performance of a general norm is affected by the performance of the norms that it generalises. Here, IRON would choose the most general norm, n_{10}, as the norm to specialise. The process would start by invoking operator `specialise`. This operator first deactivates n_{10} and activates its child norms, n_5 and n_9. Next, IRON would attempt to specialise norms n_5 and n_9. Since norm n_5 has performed well, it would remain active. However, since norm n_9 has underperformed, it would be deactivated while its children (n_7 and n_8) would be activated. Finally, the underperforming norm n_8 would be deactivated, and since it has no children (and n_7 performs well), the specialisation process would finish. The resulting network is the normative system $\Omega_2 = \{n_5, n_7\}$ in Figure 3(b).

4.7.1. Specialisation Algorithm. In Algorithm 8, the function *specialiseDown* recursively specialises a norm n into its children until it reaches the leaves of the network. If the norm is a leaf (line 1), it is deactivated so that it does not belong to the normative system (line 2). Otherwise, operator `specialise` (line 4) specialises n by (1) deactivating it and (2) activating all of its children. Next, each underperforming child norm is specialised down in turn (lines 5 through 7). As a result, the algorithm refines a generalisation by deactivating those child norms that underperformed while keeping active those that performed well. Therefore, notice that our specialisation algorithm makes it possible to perform a fine-grained backtracking over general norms that do not perform well.

ALGORITHM 8: IRON's *specialiseDown* function

Input: $n, \check{E}, \check{N}, NN, \alpha_{eff}^{spec}, \alpha_{nec}^{spec}$
Output: NN
[1] **if** *isLeaf*(n) **then**
[2] $NN \leftarrow deactivate(n, NN)$;
else
[3] $Children \leftarrow getChildren(n, NN)$;
[4] $NN \leftarrow specialise(NN, n, Children)$;
end
[5] **for** $c \in Children$ **do**
[6] **if** $underperforms(c, \check{E}, \check{N}, \alpha_{eff}^{spec}, \alpha_{nec}^{spec})$ **then**
[7] $NN \leftarrow specialiseDown(c, \check{E}, \check{N}, NN, \alpha_{eff}^{spec}, \alpha_{nec}^{spec})$;
end
end

[2]Notice that these conditions are highly conservative, as they require the higher value in the range for period T_w to be below the threshold value. This is also the case for the generalisation conditions in Equations (8) and (9), because to generalise, the lower value in the range is required to be above the threshold.

4.8. Complexity Analysis

Finally, we are ready to analyse the computational complexity of IRON. Prior to that, however, we will consider the number of norms that can be generated by a particular grammar $\mathcal{G}$. If p is the maximum number of predicates of any norm generated by the grammar, r is the maximum arity of any predicate, and d is the maximum number of terms at any position of any given predicate, the number of norms that can be generated by grammar $\mathcal{G}$ is $d^{r \cdot p}$. Given a grammar $\mathcal{G}$, we shall note the number of norms that it can generate as $\eta_{\mathcal{G}}$. Moreover, given a CBR base C, we shall note as η_C the number of norms in the case base. Now we are ready to assess IRON's complexity.

LEMMA 4.1. *The norm synthesis performed by the* IRON *algorithm when employing grammar $\mathcal{G}$ and case base C when detecting κ conflicts takes time* $O(\kappa \cdot \eta_C + 3 \cdot |Ag| \cdot |NN| + \eta_{\mathcal{G}}(|Ag| \cdot |NN| + 1))$.

PROOF. The norm generation stage involves (1) generating norms for all detected conflicts and (2) generating all potential generalisations for each new norm. The cost of the first step is $O(\kappa \cdot \eta_C)$, whereas the cost of the second step is $O(\eta_{\mathcal{G}})$. The cost of the norm evaluation process is $O(3 \cdot |Ag||NN|)$, as it involves assessing the applicability of norms ($O(|Ag||NN|)$), assessing the compliance with norms ($O(|Ag||NN|)$), and updating norms' utilities and performance ranges ($O(|Ag||NN|)$). Finally, the cost of norm refinement is $O(\eta_{\mathcal{G}} \cdot |Ag||NN|)$, which amounts to the worst-case cost of generalising norms since the cost of specialising norms is $O(|Ag||NN|^2)$. Putting it all together, the resulting worst-case time is $O(\kappa \cdot \eta_C + 3 \cdot |Ag| \cdot |NN| + \eta_{\mathcal{G}}(|Ag| \cdot |NN| + 1))$. □

Observe that the computation time of IRON is larger than BASE. Nonetheless, as we will show in the experiments carried out in Section 5, this is the price paid by IRON to significantly outperform BASE in terms of compactness.

As a final remark, notice that given a grammar $\mathcal{G}$, the number of normative systems is $2^{\eta_{\mathcal{G}}}$. This is precisely the size of the search space that IRON must explore in search of compact normative systems. However, we recall that IRON is an approximate algorithm for norm synthesis, and it does not require exploring the whole search space, as we will demonstrate in Section 5.

4.9. Architecture and Computational Model

We now have all components of the architecture and computational model of our norm synthesis system. Figure 4 illustrates the overall architecture of IRON.

As mentioned previously, IRON continuously searches online for a solution to the norm synthesis problem, namely during the operation of a NMAS. We can thus regard IRON as an external observer of agents' interactions.

IRON has a domain-independent abstract architecture that is composed of the following domain-independent elements: (1) a data structure to represent explored norms, namely a normative network; (2) a set of operators to apply changes to the normative network; and (3) a general, abstract strategy to perform norm synthesis for any scenario. To perform norm synthesis for a given scenario, this domain-independent machine receives as an input the following domain-dependent elements: (1) a function ($f_{conflict}$) to detect undesired states; (2) a grammar $\mathcal{G}$ to define norms; (3) a function to determine whether a norm applies to the agents in a given state (f_{apply}); (4) evaluation functions to compute the effectiveness (μ_{eff}) and necessity (μ_{nec}) of norms and normative systems; (5) the satisfaction degrees and thresholds (Θ); and (7) the time interval (T) considered when solving the norm synthesis problem.

Our norm synthesis mechanism is composed of (1) a normative network (NN) to compactly represent the current normative system and to store the norms synthesised (explored) so far and (2) a control unit in charge of directing norm synthesis problem

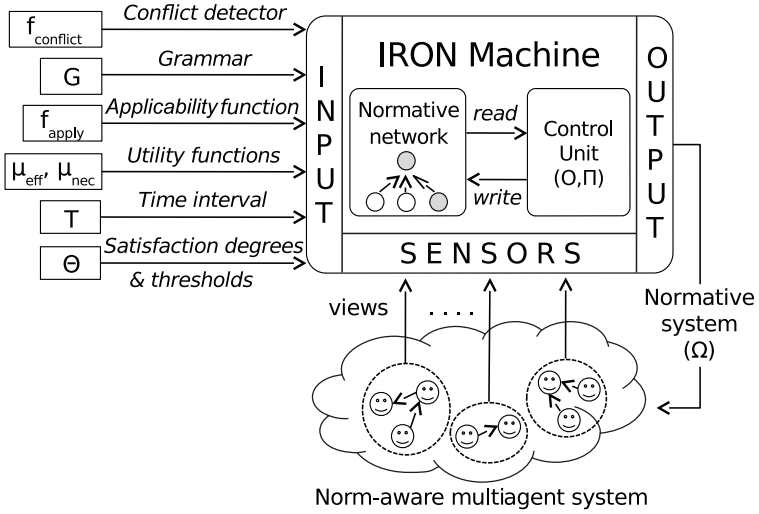

Fig. 4. IRON's architecture.

solving. The control unit continuously perceives the target NMAS by describing the current MAS state. After that, the control unit calls the strategy Π described in Section 4.5 to apply a collection of operators O (see Section 4.4) and to eventually produce a new normative system that prevents the conflicts observed in the current system state.

The normative system (Ω) is broadcast to the agents in the NMAS. Once the new normative system is deployed, the control keeps on perceiving the MAS and generating descriptions of states of the NMAS to be analysed by the strategy. This cyclic process continues until the control unit receives from the strategy a normative system that is evaluated effective and necessary enough, according to the evaluation functions and satisfaction degrees set as input, during a period of time T. Such a normative system will represent a solution to the norm synthesis problem.

5. EMPIRICAL ANALYSIS AND RESULTS

In this section, we present an empirical evaluation of the IRON norm synthesis mechanism. Section 5.1 describes the experimental settings, and Section 5.2 provides results of the empirical evaluation; Sections 5.2.1 and 5.2.2 illustrate a stability analysis and a convergence analysis of IRON, respectively. Finally, Section 5.2.3 compares IRON with BASE in terms of stability and compactness. We demonstrate that IRON is more stable than BASE and synthesises normative systems that are more compact than those synthesised by BASE.

5.1. Empirical Settings

With the aim of comparing IRON and BASE, our experiments use the same traffic scenario than the one used in Morales et al. [2011c]. Our experiments simulate a traffic junction composed of two orthogonal roads represented by a 20×20 grid (simulator available in the appendix). Each road has two 20-cell lanes (one per direction).

Figure 5(a) shows a 12×12 subgrid that corresponds to the centre of the junction. Each agent is a car that travels along the grid at one cell per tick by following a random trajectory. Specifically, each car enters the scenario from four possible start points (light/green points in Figure 5(a)) and travels towards randomly chosen destinations (exit points, depicted in dark/red in Figure 5(a)). In this particular scenario, undesired states are those MAS states that contain collisions between cars. Thus, IRON will synthesise norms to avoid collisions. Since IRON is devoted to avoid undesired states, we consider a scenario that potentially leads to a large number of undesired states.

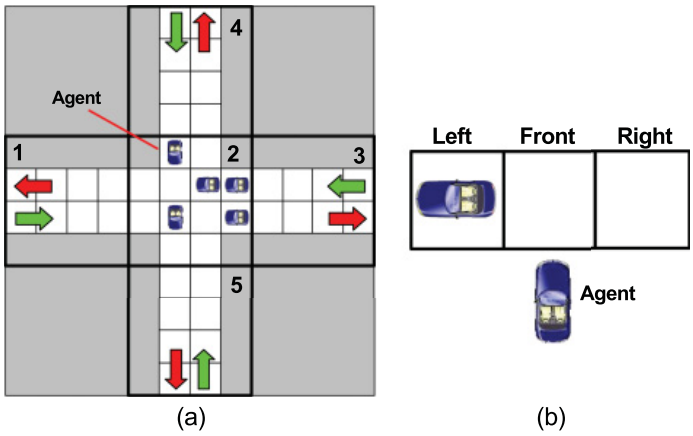

Fig. 5. (a) Central area of the traffic junction. (b) Agent context.

Therefore, to favour a high frequency of collisions (i.e., undesired states), we use a high traffic density (from 41% to 48% of occupied cells) by having three cars entering the scenario at every tick. At each tick, each car decides whether to comply or not with the norms output by IRON according to a probability, namely a *norm infringement rate*. The norm infringement rate is fixed at the beginning of each simulation and is the same for all cars.

Each experiment consists of a set of different simulations. IRON starts each simulation with an empty normative system.[3] As the simulation progresses, collisions among cars occur, and IRON then synthesises normative systems to avoid future collisions. The simulation finishes whenever it reaches 50,000 ticks or IRON converges to a stable normative system, hence removing undesired states and solving the norm synthesis problem. We assume that IRON has converged to a normative system if during a 10,000-tick period (1) the normative system remains unchanged and (2) no new (nonregulated) undesired MAS states are detected. Nonregulated conflicts are those undesired MAS states that have not arisen from previous norm infringements. That is, no agent infringed on a norm during the transition from the previous state to the undesired state. We assume that an undesired state that arises from a norm infringement is a regulated conflict (i.e., a previously arisen undesired state that triggered the generation of the currently infringed norm), and thus it is not considered when assessing convergence. This assumption is based on the intuition that we cannot ensure that if the agents had fulfilled the norm, the undesired state would have arisen.

To generate norms, IRON receives a grammar as an input as follows. The precondition of each norm contains three predicates *left*, *front*, *right*, which represent the three cells in the context of a reference car (see Figure 5(b)). Each predicate contains one term out of a set of six terms {*car-heading-left*, *car-heading-right*, *car-opposite-heading*, *car-same-heading*, *nil*, *anything*.} The first four terms represent a car along with the direction in which it is heading, whereas term *nil* represents no car and term *anything* represents both a car or nothing. Furthermore, the following subsumption relationships for the terms in the grammar hold: *car-heading-left* $\sqsubseteq$ *anything*, *car-heading-right* $\sqsubseteq$ *anything*, *car-opposite-heading* $\sqsubseteq$ *anything*, *car-same-heading* $\sqsubseteq$ *anything*, *nil* $\sqsubseteq$ *anything*. The postcondition of each norm uses only prohibitions ($\theta = prh$) over an action. The actions available to agents are $Ac = \{Go, Stop\}$. Nevertheless, in this particular setting, car agents just perform action *Stop* whenever a norm prohibits them from going forward. Therefore, the grammar that we employ can

[3]Notice, however, that IRON can also start operating with a nonempty normative system.

synthesise $6^3 = 216$ different norms, and the number of normative systems to consider amounts to 2^{216} ($> 10^{65}$). As an example, in our experiments, IRON generates norms of the following form:

$$n : \langle \{left(car\text{-}heading\text{-}right), front(car\text{-}heading\text{-}right), right(car\text{-}heading\text{-}right\}, prh(Go)\rangle$$

$$n' : \langle \{left(car\text{-}heading\text{-}right), front(car\text{-}heading\text{-}right), right(nil)\}, prh(Go)\rangle.$$

Norm n prohibits a car from moving on (hence giving way) if the three cells in its context contain cars, each heading towards its right. Similarly, norm n' prohibits a car from going if the left and front cells in its context contain cars heading towards its right and its right cell contains nothing. Notice that both norms n, n' only differ in their right position.

Regarding IRON's configuration parameters, we have taken a conservative approach to set them. This decision is intended to refine the normative system only when norms are slightly effective or necessary. We set IRON's parameters as follows: (1) low deactivation thresholds ($\alpha_{eff}^{spec} = 0.2, \alpha_{nec}^{spec} = 0.2$) to only deactivate norms performing very poorly, (2) high generalisation thresholds ($\alpha_{eff}^{gen} = 0.6, \alpha_{nec}^{gen} = 0.4$) to only generalise norms when performing very well, (3) weights in Equation (3) $w_{A_C} = 5$ and $w_{A_{\bar{C}}} = 1$ to ensure that norm fulfilments leading to collisions (ineffective norms) are much more penalised than those avoiding collisions (effective norm), and (4) weights in Equation (5) $w_{H_I} = 2$ and $w_{S_I} = 1$ to ensure that norm infringements leading to collisions (harmful infringements, necessary norm) obtain a much higher reward than those leading to no collisions (successful infringements, unnecessary norm). Additionally, we compute norms' effectiveness and necessity range over a long period of $|T_w| = 200$ different norm evaluations, and norms effectiveness and necessity values, namely μ_{eff}, μ_{nec}, are initially set to a value $k = 0.5$.

5.2. Empirical Results

We now analyse the results of our empirical evaluation. First, we perform a stability analysis of IRON. We perform a microanalysis to study how IRON manages to solve the norm synthesis problem, hence removing new collisions (i.e., nonregulated collisions) in our traffic scenario. Additionally, we perform a macroanalysis showing that IRON is able to converge despite a large proportion of noncompliant behaviours in the overall agent population. Second, we compare IRON with BASE in terms of stability and compactness. We show that IRON is more stable than BASE, as it converges for greater norm infringement rates than BASE. Moreover, IRON manages to synthesise more compact normative systems than those synthesised by BASE, decreasing the overall number of norm predicates in the normative system.

5.2.1. Stability Analysis of IRON. We present a stability analysis to show how IRON manages to successfully synthesise normative systems that solve the norm synthesis problem. We compute the stability degree as a convergence rate, namely the number of simulations that converged to a stable normative system out of the total number of simulations. We notice that since undesired states trigger norm synthesis, the lack of collisions causes the normative system to remain stable.

We performed a simulation to show IRON's convergence process. In particular, Figure 6 illustrates the normative changes (i.e., the timesteps in which the the normative network and/or the normative system change) for a single simulation with 0.3 norm infringement rate. That is, on average, agents decide to infringe norms 3 out of 10 times, which can be considered as a high enough norm infringement rate. This figure shows the following: (1) the average of new car collisions (i.e., collisions that are not caused by norm infringements) per tick along time; (2) the normative network

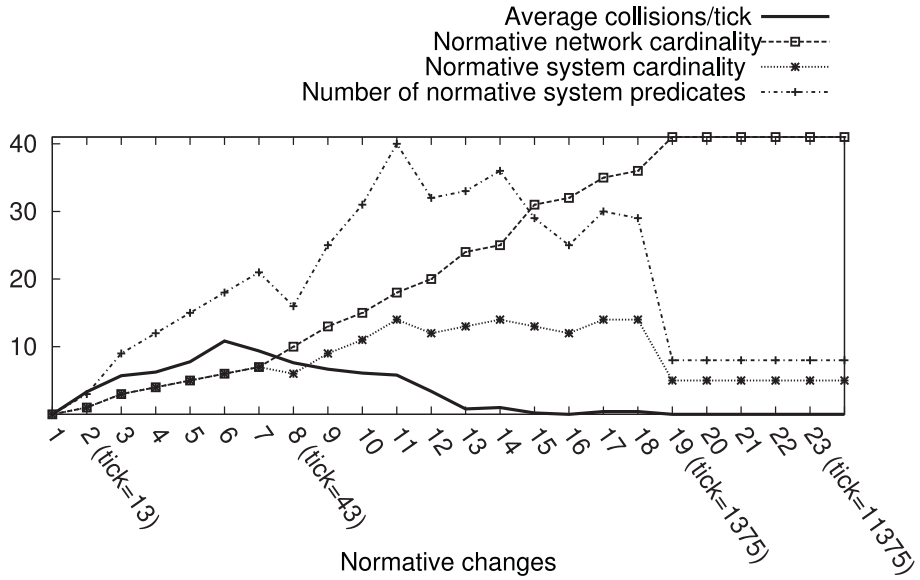

Fig. 6. Norm synthesis along a single simulation. The x -axis corresponds to normative changes (i.e., changes in the normative network and/or the normative system), and the y -axis corresponds to the different measures in the legend.

cardinality, namely the total number of synthesised norms; (3) the cardinality of the normative system, namely the number of active norms in the normative network; and (4) the number of predicates in the normative system. At tick 13 (which corresponds to the 2nd normative change), the first collision arises and IRON synthesises the first norm. From that tick onwards, IRON keeps generating norms when needed, hence increasing the cardinality of both the normative network and the normative system. As a consequence, the number of predicates in the normative system increases as well. At tick 43 (8th normative change), IRON performs the first norm generalisation, reducing the cardinality of the normative system from 9 to 8 norms. As a result of this norm generalisation, the number of norms and predicates in the normative system decreases, thus increasing its compactness. Up to tick 1,375 (19th normative change), IRON keeps generating and generalising norms when possible. Norm generalisations reduce the total number of predicates of the normative system. At tick 1,375, IRON performs the last norm generalisation, hence synthesising a compact normative system of 5 norms with 8 predicates in total. From tick 1,375 onwards, the normative system remains stable. By using the resulting normative system, cars that comply with norms do not cause collisions. However, those cars may collide with other cars that infringe on norms. Recall that those collisions arising from norm infringements are not taken into account when assessing convergence. After 10,000 further ticks, IRON reaches the convergence criteria (tick 11,375/23th normative change). Overall, IRON explored 41 different norms (out of 125 possible ones), which were generalised into 5 norms, to find a 5-norm normative system that successfully prevents collisions as long as cars comply with norms.

The five-norm normative system to which IRON converged is shown in Table III. Norm n_1 is a left-hand side priority norm specifying that a car must stop when it observes a car to its left that is heading to its right, and no matter what it perceives in front or to its right. It has very high values of effectiveness (0.86) and necessity (0.90), as it represents a situation that leads to collisions most of the time. Therefore, we consider this norm to be an *essential* norm. On the other hand, norm n_4 forces a car to stop when it observes a car in front heading in the same direction of the reference car. This situation rarely leads to collisions, as the car in front rarely stops. As a consequence, its

Table III. A Normative System upon Convergence

Norm	Precondition (θ)	Modality	$\bar{\mu}_{eff}$	$\bar{\mu}_{nec}$
n_1	*left*(*car-heading-right*)	*prh*(*Go*)	0.86	0.90
n_2	*left*(*car-heading-left*), *front*(*car-heading-left*)	*prh*(*Go*)	0.87	0.73
n_3	*front*(*car-heading-right*), *right*(*car-heading-right*)	*prh*(*Go*)	0.86	0.81
n_4	*front*(*car-same-heading*)	*prh*(*Go*)	0.83	0.33
n_5	*front*(*car-heading-left*), *right*(*car-heading-left*)	*prh*(*Go*)	0.81	0.75

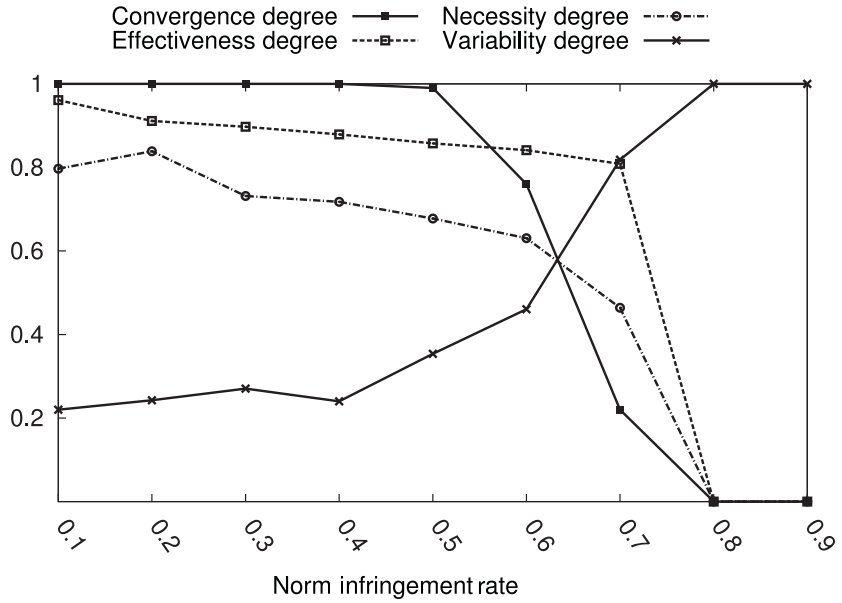

Fig. 7. Robustness analysis depending on norm infringements. The x-axis represents the different norm infringement rates, and the y-axis represents the different degrees that are given in the legend.

average necessity ($\bar{\mu}_{nec}$) has a low value (0.33). We say then that this norm is *preventive*, as agents should comply with it to avoid collisions that rarely occur.

We have thus shown that IRON can successfully synthesise a compact normative system with high effectiveness.

5.2.2. Macro analysis. We now explore the limits of IRON by testing its synthesis capabilities under different norm infringement rates. Specifically, we analyse IRON's percentage of convergence for different norm infringement rates, ranging from 0.1 to 0.9. We performed 100 simulations per norm infringement rate. Figure 7 shows averaged results for the effectiveness and necessity of the synthesised normative systems.[4]

Moreover, the *convergence degree* series shows the number of runs that converged to a stable normative system out of the 100 runs. For low and medium norm infringement rates (up to 0.4), IRON's stability degree is 1, namely it successfully converged 100% of the times. Furthermore, it converged to normative systems with high effectiveness (0.88) and necessity (0.71). Between medium and high norm infringement rates (0.4 and 0.7), the convergence decreased (due to oscillations in the normative systems), and it is for very high norm infringement rates (from 0.8 onwards, which means that agents decide to infringe on norms 8 or more times out of 10 times) that IRON cannot find a normative system. Overall, IRON proved to be highly resilient to noncompliant behaviours during

[4]For the sake of clarity, we do not plot standard deviations. However, it is worth mentioning that the standard deviations for effectiveness and necessity for each norm infringement rate are within [0.006, 0.011] and [0.080, 0.0137], respectively.

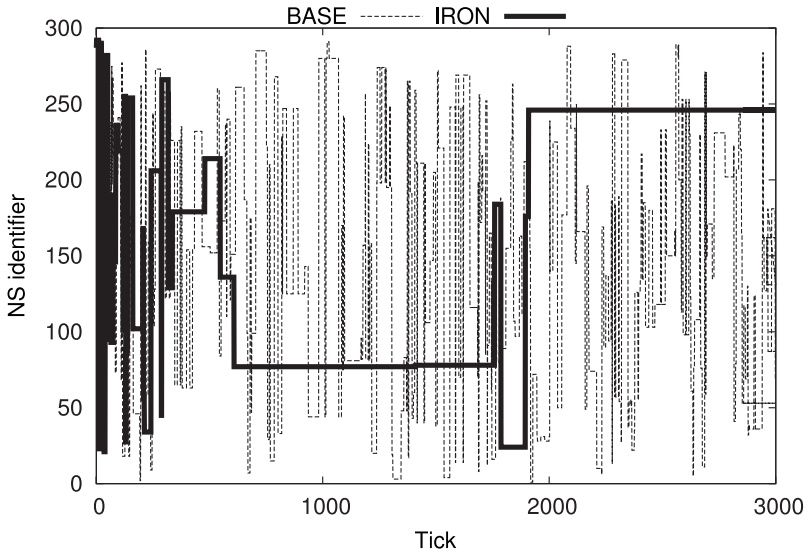

Fig. 8. Normative system changes along time.

the synthesis process. IRON managed to successfully synthesise norms despite up to 40% norm infringement rate of agents.

Figure 7 also shows the variability of IRON's synthesis, namely whether it yields different normative systems. Below the 0.4 norm infringement rate, the variability remained near 0.2 (i.e., 100 executions converged to 20 different normative systems). Since preventive norms become unstable (activated and deactivated back and forth) with high norm infringement rates, IRON takes longer to synthesise stable norms. This leads to new, different normative systems, which IRON did not need to explore with lowernorm infringement rates.

5.2.3. IRON versus BASE: Comparison of Stability and Compactness. Next we compare IRON with BASE, the mechanism for norm synthesis described in Section 3. Our experiments employ the publicly available version of BASE at Morales et al. [2011b]. Our comparison employs the same traffic scenario described in Section 5.1 and the experimental settings for BASE described in Morales et al. [2011c].

Stability degree. We first compare both norm synthesis approaches in terms of their stability. We performed 100 simulations per norm synthesis method. Our analysis is performed for very low (0.1), low (0.2), medium (0.3 to 0.6), and high (beyond 0.6) norm infringement rates. Figure 8 illustrates how normative systems change along a single, sample simulation: the switch frequency between different normative systems of BASE is much higher than that of IRON, which stabilises normative systems for longer periods of time until it converges. At the end of the simulation, BASE explored 251 different normative systems and was not able to converge, whereas IRON explored 41 different normative systems, converging to a final normative system. Additionally, Figure 9 compares the stability degree of both norm synthesis methods over 100 simulations. For very low norm infringement rates (0.1), both methods successfully converge to a normative system that effectively coordinates the MAS. Nevertheless, beyond low norm infringement rates (0.2), BASE dramatically decreases its stability degree (i.e., it becomes more inefficient in synthesising a normative system that avoids collisions). In fact, it never manages to converge to a normative system beyond the 0.3 norm infringement rate, and collisions are never completely eradicated, hence leading to a 0 stability degree. As for IRON, it converges for low norm infringement rates and totally removes collisions 100% of the time. Still, for medium norm infringement rates (0.3

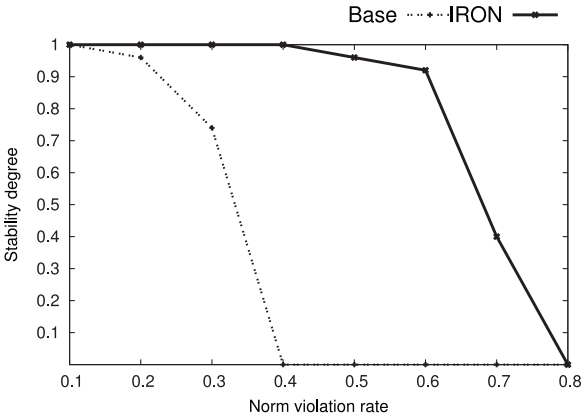

Fig. 9. Comparison of the stability degree of IRON and BASE.

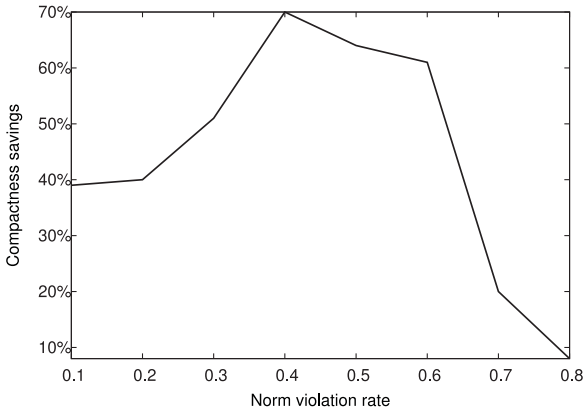

Fig. 10. Compactness savings of IRON with respect to BASE.

to 0.6), its stability degree is between 0.9 and 1, namely it converges between 90% and 100% of the simulations. The stability degree of IRON tends to decrease beyond high norm infringement rates (0.6), and it is for very high norm infringement rates (0.8) that it fails to converge. Overall, IRON is much more stable than BASE, allowing convergence for much higher norm infringement rates.

Compactness savings. We now compare the compactness of the normative systems synthesised by IRON and BASE. Recall from Definition 5 that we measure the compactness of a normative system in terms of its overall number of norm predicates. Figure 10 illustrates the compactness savings achieved by IRON with respect to BASE: it saves compactness for all norm infringement rates, achieving its best results for medium norm infringement rates.

For low norm infringement rates (up to 20% norm infringement rates), IRON manages to converge to normative systems that are between 30% and 40% more compact (have fewer norms) than those synthesised by BASE. As for medium norm infringement rates (0.3 to 0.6), IRON obtains its best savings in compactness (up to 70%: from 36.2 down to 10.5 predicates on average). Here, IRON benefits from its stability, whereas BASE is penalised by its instability. This is because when BASE does not converge, it typically outputs normative systems with a low compactness. Specifically, whereas IRON manages to converge between 90% and 100% of the time, BASE just converges 72% of the time for a norm infringement rate of 0.3, and it never converges beyond a 0.4 norm infringement rate. As a result, IRON synthesises normative systems that are up to 70% more compact than those synthesised by BASE. As for high norm infringement rates (beyond 0.6), IRON's savings in compactness tend to decrease since its percentage of convergence decreases as well. Overall, IRON synthesises normative systems that are much more compact than those synthesised by BASE, allowing agents to save computational resources when processing norms.

6. CONCLUSIONS AND FUTURE WORK

In this article, we have introduced IRON, an approach for the online synthesis of norms for MAS. IRON is a domain-independent architecture and computational model composed of three abstract elements: (1) a normative network to represent the norms generated during the synthesis process, (2) a set of operators to apply changes to the normative network, and (3) a strategy to perform norm synthesis. One can use IRON to perform norm synthesis on different domains by implementing relatively simple domain-dependent functions such as conflict detection and norm applicability.

We have performed an empirical evaluation of IRON on a simplified traffic junction scenario. In our scenario, cars are autonomous agents that drive through a junction to reach their destinations. The goal of IRON is to synthesise norms that prevent cars from colliding. Our empirical results show that IRON significantly outperforms BASE, per the norm synthesis techniques presented in Morales et al. [2011c]. On the one hand, we have shown that IRON's normative systems are up to 70% more compact than those generated by BASE. On the other hand, we have also shown that unlike BASE, IRON successfully synthesises normative systems despite high norm infringement rates. These improvements mainly derive from the fact that IRON takes more informed and fine-grained decisions than BASE. This comes at the cost of employing (1) a generalisation function that requires complete evidence prior to performing norm generalisations and (2) a specialisation function that allows performing a fine-grained backtracking of norm generalisations.

As future work, there are multiple opportunities for research. First, we plan to enrich our current scenario to include further traffic elements like pedestrians, cars driving at different speeds, or even traffic lights. Along this line, in Morales et al. [2011a], we already performed an empirical evaluation that compared how BASE and traffic lights managed to regulate a traffic junction. We believe that it is worth pursuing this line of inquiry. Second, we plan to enhance IRON to synthesise norms while considering multiple goals at the same time, instead of a single goal as we do in this article. For instance, norms could be synthesised also considering the traffic flow. That is, IRON could synthesise norms to (1) avoid car collisions and (2) to avoid traffic jams. Notice that this setting is an especially interesting topic of study, as both goals are contradictory. On the one hand, avoiding car collisions requires occasional car stops, which is prejudicial for the traffic flow. On the other hand, undoing a traffic jam requires cars to start moving, which can lead to occasional collisions. Third, and finally, we plan to investigate how to further improve compactness. This is motivated by our analysis of the normative systems synthesised by IRON. We have observed that several norms may apply to the very same state. In particular, we have found two new types of relationships that can be characterised as follows: (1) two norms apply to a state, but only one of them suffices to regulate the state (the other one is therefore superfluous), and (2) two norms apply to a state, but they perform (in terms of effectiveness) worse together than separately. The first type of relationship is directly related to overregulation: regulating a state with two norms, when only one of them is sufficient, implies overconstraining the behaviour of agents. Thus, removing one of them may decrease overregulation, hence increasing individual agents' freedom. The second type of relationship captures a "conflict" between norms: the two of them together in the normative system decrease the effectiveness of a normative system. From this discussion follows that capturing these two relationships will help us improve compactness and, eventually, effectiveness.

7. DISCUSSION

In what follows, we focus on providing pragmatic guidelines to help potential users decide *when* and *how* to employ IRON.

First, we analyse the requirements that a scenario, besides the traffic scenario described in this article, must satisfy to apply IRON:

—The norm generation approach employed by IRON is based on CBR, and thus it assumes that similar problems (in our case, conflicts) have similar solutions (norms).
—MAS conflicts must be identifiable. Since IRON's norm synthesis is based on the detection of undesired states, it must be able to decide whether a state represents a conflict or not.

—Agents' actions must be observable, and in case they cause conflicts, these arise immediately. IRON currently assumes that whenever a conflict arises at time t, the *cause* of the conflict can be found at the action of an agent at previous timestep $t-1$. Moreover, IRON also requires the agent causing the conflict to be involved in it at time t.
—The number of agents involved in a conflict must be limited. Before IRON generates a norm that successfully regulates a conflict, it may generate a different norm for each of the agents involved in it. This approach is only feasible if the number of agents is limited, and hence the number of possible norms to generate for a given conflict is limited as well.
—The number of norms that the grammar employed by IRON can generate must not be large. Recall that the computational complexity of IRON's synthesis is linear to the number of norms that a grammar can generate, but this number is exponential to the number of predicates and their arity.

In this work, we have used a simplified traffic scenario to illustrate IRON. However, we argue that IRON can be employed in a vast variety of scenarios whenever the preceding conditions hold. As an example, in Morales et al. [2014], we show how to apply IRON to perform norm synthesis in an online community scenario. In that scenario, the users of the community upload and view contents, and they complain about contents that they find inappropriate for the community (e.g., spam or porn contents). There, undesired states are those where a significant amount of users complain about a given content. We showed that IRON is capable of synthesising norms that prevent users from uploading conflicting contents.

Second, we focus on how to tune the parameters employed by IRON. The guidelines that follow stem from a series of experiments that we conducted to understand the impact of the specialisation and generalisation thresholds on IRON 's convergence. More precisely, we ran the empirical evaluation described in Section 5, but this time combined different specialisation thresholds (low, medium, and high values of α_{eff}^{spec}, α_{nec}^{spec}) with different generalisation thresholds (low, medium, and high values of α_{eff}^{gen}, α_{nec}^{gen}).[5] Two main observations derive from our analysis:

—The value of IRON's specialisation (deactivation) thresholds directly impact on the type of norms that become part of synthesised normative systems. On the one hand, high specialisation thresholds lead IRON to include in a normative system those norms that are highly effective and necessary. Therefore, those norms that regulate situations that occasionally lead to conflicts (i.e., that are slightly necessary) are never included. Thus, we should use high specialisation thresholds when we aim at very compact normative systems and are ready to tolerate occasional conflicts. On the other hand, low specialisation thresholds lead IRON to include in a normative system any norm that avoids undesired states, even though the situation it regulates very sporadically leads to conflicts.
—The value of IRON's generalisation thresholds affect the compactness of synthesised normative systems. On the one hand, high generalisation thresholds mean that only those norms that are highly effective and necessary can be generalised. Since performing norm generalisations is costly, this setting is appropriate in scenarios where the cost of generalising norms is relevant. Thus, this setting guarantees that a norm is only generalised when it performs well. On the other hand, low generalisation

[5]For the sake of keeping the article length within reasonable limits, we did not incorporate this new set of experiments with different threshold values, but we do report on our main findings.

thresholds allow IRON to generalise a norm even though its effectiveness and necessity values are low. This leads to increasing the compactness of the normative system.

REFERENCES

Agnar Aamodt and Enric Plaza. 1994. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *Artificial Intelligence Communications* 7, 1, 39–59.

Cristina Bicchieri. 2006. *The Grammar of Society: The Nature and Dynamics of Social Norms*. Cambridge University Press.

Ken Binmore. 2005. *Natural Justice*. Oxford University Press.

Guido Boella, Leendert van der Torre, and Harko Verhagen. 2006. Introduction to normative multiagent systems. *Computational and Mathematical Organization Theory* 12, 2–3, 71–79.

George Christelis and Michael Rovatsos. 2009. Automated norm synthesis in an agent-based planning enviroment. In *Proceedings of the 8th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'09)*. 161–168.

Frank Dignum. 1999. Autonomous agents with norms. *Artificial Intelligence and Law* 7, 1, 69–79.

David Fitoussi and Moshe Tennenholtz. 1998. Minimal social laws. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*. 26–31.

Andrés García-Camino, Juan A. Rodríguez-Aguilar, Carles Sierra, and Wamberto Vasconcelos. 2009. Constraint rule-based programming of norms for electronic institutions. *Autonomous Agents and Multi-Agent Systems* 18, 1, 186–217.

Nathan Griffiths and Michael Luck. 2010. Norm emergence in tag-based cooperation. In *Proceedings of the 9th International Workshop on Coordination, Organization, Institutions, and Norms in Multi-Agent Systems (COIN'10)*. 80–87.

James E. Kittock. 1993. Emergent conventions and the structure of multi-agent systems. In *Lectures in Complex Systems: The Proceedings of the 1993 Complex Systems Summer School* (Lecture Vol. VI), L. Nadel and D. Stein (Eds.). Addison-Wesley, 507–521.

John-Jules Ch. Meyer, and Roel J. Wieringa. 1993. *Deontic Logic in Computer Science: Normative System Specification*. John Wiley and Sons, Chichester, UK.

Javier Morales, Maite López-Sánchez, and Marc Esteva. 2011a. Evaluation of an automated mechanism for generating new regulations. In *Proceedings of the 14th International Conference on Advances in Artificial Intelligence: Spanish Association for Artificial Intelligence (CAEPIA'11)*. 12–21.

Javier Morales, Maite López-Sanchez, and Marc Esteva. 2011b. Source code of using experience to generate new regulations. Retrieved January 28, 2014, from www.iiia.csic.es/~jmorales/Downloads/Morales_IJCAI2011_SourceCode.tar.gz.

Javier Morales, Maite López-Sánchez, and Marc Esteva. 2011c. Using experience to generate new regulations. In *Proceedings of the International Joint Conference in Artificial Intelligence (IJCAI'11)*. 307–312.

Javier Morales, Maite López-Sanchez, Juan A. Rodríguez-Aguilar, Michael Wooldridge, and Wamberto Vasconcelos. 2013. Automated synthesis of normative systems. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'13)*. 483–490.

Javier Morales, Iosu Mendizábal, David Sánchez-Pinsach, Maite López-Sánchez, and Juan A. Rodríguez-Aguilar. 2014. Using IRON to build frictionless on-line communities. *AI Communications: Artificial Intelligence in the Catalan Association for AI* 28, 1, 55–71.

Stephen Muggleton and Luc De Raedt. 1994. Inductive logic programming: Theory and methods. *Journal of Logic Programming* 19, 20, 629–679.

Shan-Hwei Nienhuys-Cheng and Ronald de Wolf. 1997. *Foundations of Inductive Logic Programming*. Springer-Verlag, New York, NY.

Christopher K. Riesbeck and Roger C. Schank. 1989. *Inside Case-Based Reasoning*. Lawrence Erlbaum Associates, Hillsdale, NJ.

Norman Salazar, Juan A. Rodriguez-Aguilar, and Josep L. Arcos. 2010. Robust coordination in large convention spaces. *AI Communications* 23, 4, 357–372.

Onkur Sen and Sandip Sen. 2010. Effects of social network topology and options on norm emergence. In *Proceedings of the 5th International Conference on Coordination, Organizations, Institutions, and Norms in Agent Systems (COIN'09)*. 211–222.

Sandip Sen and Stéphane Airiau. 2007. Emergence of norms through social learning. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07)*. 1507–1512.

Yoav Shoham and Kevin Leyton-Brown. 2009. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, New York, NY.

Yoav Shoham and Moshe Tennenholtz. 1995. On social laws for artificial agent societies: Off-line design. *Journal of Artificial Intelligence* 73, 1–2, 231–252.

Munindar P. Singh, Matthew Arrott, Tina Balke, Amit K. Chopra, Rob Christiaanse, Stephen Cranefield, Frank Dignum, Davide Eynard, Emilia Farcas, Nicoletta Fornara, Fabien Gandon, Guido Governatori, Hoa Khanh Dam, Joris Hulstijn, Ingolf Krueger, Ho-Pun Lam, Michael Meisinger, Pablo Noriega, Bastin Tony Roy Savarimuthu, Kartik Tadanki, Harko Verhagen, and Serena Villata. 2013. The uses of norms. In *Normative Multi-Agent Systems*, G. Andrighetto, G. Governatori, P. Noriega, and L. W. N. van der Torre (Eds.). Dagstuhl Follow-Ups, Vol. 4. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 191–229.

Richard S. Sutton and Andrew G. Barto. 1998. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA.

Daniel Villatoro, Jordi Sabater-Mir, and Sandip Sen. 2011. Social instruments for robust convention emergence. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI'11)*. 420–425.

Adam Walker and Michael Wooldridge. 1995. Understanding the emergence of conventions in multi-agent systems. In *Proceedings of the International Conference on Multiagent Systems (ICMAS'95)*. 384–389.

Chao Yu, Minjie Zhang, Fenghui Ren, and Xudong Luo. 2013. Emergence of social norms through collective learning in networked agent societies. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'13)*. 475–482.

Received December 2013; revised November 2014; accepted January 2015