# Using IRON to Build Frictionless On-line Communities

Javier Morales [a,b], Iosu Mendizabal [b], David Sanchez-Pinsach [b],
Maite Lopez-Sanchez [b] and Juan A. Rodriguez-Aguilar [a]
[a] *Artificial Intelligence Research Institute, IIIA-CSIC, Spanish National Research Council.*
*E-mail: {jmorales, jar}@iiia.csic.es*
[b] *MAiA Department, Universitat de Barcelona, Spain.*
*E-mail: iosu.mendizabal@alumni.mondragon.edu, sdividis@gmail.com, maite@maia.ub.es*

On-line communities are virtual environments where users exchange contents. Occasionally, users' interactions lead to frictions, jeopardising the proper functioning of the community. Trying to avoid frictions, on-line communities typically incorporate a regulation mechanism based on (i) norms set by the owner of the community; and (ii) human moderators. In this paper we present a *participatory legislation mechanism* that automatically synthesises norms for an on-line community based on users' complaints about contents. With this aim, we present an agent-based simulator to model the interactions within on-line communities. We then exploit IRON, an automatic norm synthesis mechanism, to regulate simulated on-line communities. As a result, IRON synthesises norms that prevent a user from uploading contents that users regard as unacceptable by means of complaints, hence avoiding frictions.

Keywords: multi-agent normative systems, on-line communities, norm synthesis

## 1. Introduction

Since the diffusion of the Internet in the middle 90's, on-line communities have become extremely popular tools for content exchange. On-line communities are virtual environments that work over the Internet, allowing users to share different types of contents (e.g., opinions, videos, and pictures). An on-line community is an open, dynamic system. On the one hand, it is open because the users may enter and exit along time. On the other hand, it is dynamic because users may also change their preferences and behaviours. Within on-line communities users eventually interact with other users that have very different opinions. As a consequence, occasionally users' interactions may lead to frictions between users, which can be regarded as conflicting situations (i.e., conflicts). These situations complicate information exchange, causing discomfort to users, who eventually may abandon the community. As an example, uploading an inappropriate content can be regarded as a conflict. Consider a user that uploads a content that is not considered as acceptable by other users, hence leading these to complain about it. Users may engage in an argument, hence jeopardising the frictionless of the on-line community.

Most on-line communities (e.g., Facebook, Twitter) regulate users' behaviours to prevent users from uploading contents that cause users' complaints, hence ensuring frictionless interactions. Typically, a regulation mechanism is based on: (i) some pre-designed *terms and conditions* (i.e., norms) set by the owner of the community that describe, in general terms, users' obligations, permissions and prohibitions; and (ii) human *moderators* who actively participate in the community, performing corrective actions in order to avoid and solve conflicts. In some way, moderators' actions can be regarded as norms that are not explicit for the users. However, this regulatory approach suffers from lack of:

1. *Democracy.* The terms and conditions of an on-line community are fixed by the owner of the community, hence not allowing users to participate in the regulation process.
2. *Specificity.* From a user's point of view, the norms of the community are not clear enough. The terms and conditions of on-line communities are general descriptions of what users should and should not do, but in general they are not specific enough.
3. *Transparency.* The corrective actions that moderators perform are somehow subjective, since they de-

AI Communications
ISSN 0921-7126, IOS Press. 

pend on their individual beliefs and principles. As an example, consider a user that shares spam contents within a community. A particular moderator may decide to remove the content and ban the user for three days, while another moderator may decide to simply remove the content.
4. *Adaptability.* Users and their behaviours may change along time. Therefore, the regulations that initially performed well to regulate users' behaviours may not be effective any longer as the community changes.

Against this background, the main contribution of this paper is a *participatory legislation mechanism* for on-line communities aimed at trying to ensure frictionless users' interactions. Our mechanism takes into account users' actions and feedback to synthesise norms that prevent users from uploading inappropriate contents, hence avoiding frictions. Thus, the regulation of the community becomes a participatory process involving users. With this aim, our mechanism is founded on the configuration and exploitation of IRON (Intelligent Robust On-line Norm synthesis machine) [12], a state-of-the-art norm synthesis mechanism, to employ it in an agent-based simulation of an on-line community. More precisely, the contributions of this paper are:

1. *An On-line Communities Simulator* that allows to simulate an on-line community as a Multi-Agent System (MAS). In our simulator, users are modelled as agents that interact by exchanging contents.
2. *The adaptation of* IRON [12] to synthesise norms for our on-line community simulator at runtime. In this work, we exploit IRON to synthesise norms aimed at avoiding frictions, considering users' actions and feedback.
3. *An empirical evaluation* that shows that IRON is capable to synthesise norms for the users of our simulated on-line community, based on the complaints they report about conflicting contents. In general, we observe that regulation occurs when the population as a whole produces a significant enough number of complaints.

As a benefit of employing IRON, our mechanism manages to synthesise norms which are: (i) concise, allowing to regulate a MAS with a minimal set of norms; and (ii) specific, since they describe users' obligations and prohibitions for specific situations; Moreover, our resulting legislation mechanism is: (i) participatory, since the users of the community participate in the synthesis of norms by means of their complaints; (ii) transparent, because users are provided with explicit norms; and (iii) adaptive, since IRON is capable of adapting the norms of the community to changes.

To the best of our knowledge, this paper is the first contribution that studies the automated synthesis of norms for on-line communities.

The paper is organised as follows. Next Section 2 describes our on-line community simulator. Section 3 describes IRON mechanism so that Section 4 can detail the configuration and exploitation of IRON to synthesise norms for our on-line community scenario. Section 5 illustrates an empirical evaluation of IRON's norm synthesis for our on-line community scenario. Section 6 connects this work with related work. Finally, Section 7 draws some conclusions and sets paths to future research.

## 2. A Simulator for On-line Communities

Our On-line Communities Simulator allows to perform agent-based time-discrete simulations of users' interactions within an on-line community scenario. It has been implemented with the aid of *Repast Simphony* [13]. In what follows we outline its features.

### *2.1. Simulator Outline*

Within an on-line community, each user's behaviour is simulated by means of an agent. Henceforth we will refer to user and agent interchangeably. The on-line community has a population of users that interact by exchanging contents in three different *sections*: `The Reporter`, `Forum` and `Multimedia`. Section `The Reporter` allows users to share news. In the `Forum` section, users interact by sharing contents. Finally, section `Multimedia` lets users to upload multimedia contents such as videos and pictures.

Briefly, our simulator works as follows. Time is discrete and measured in *ticks*. At each tick, each community user may: (1) *upload* new contents to some section in the community; (2) *view* contents; and (3) *complain* about viewed contents that she regards as conflictive. Each content a user uploads has a category out of a set of categories $C_{ct} = \{\texttt{correct}, \texttt{spam}, \texttt{porn}, \texttt{insult}, \texttt{violent}\}$. On the one hand, `correct` contents are those that users feel comfortable with, and hence do not complain about. On the other hand, conflicting contents (i.e., `spam`, `porn`, `insult` and `violent`) are those that may cause frictions between users. For the sake of simplicity, in our simulator agents use a contents database in order to generate contents. Furthermore, we assume that contents in the database are previously

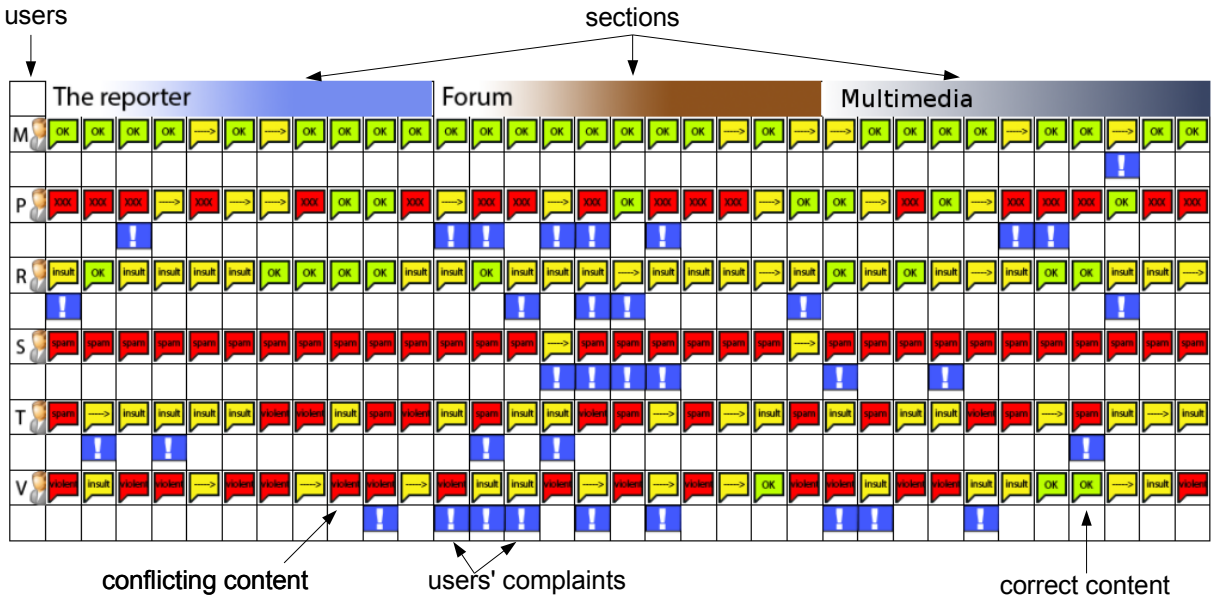



Fig. 1. On-line Communities Simulator: Grid representation of our on-line community scenario.

assigned a category out of $C_{ct}$, and that the assigned category is correct.

Our simulator allows to classify community users with different behaviours regarding which categories of contents they upload. In particular, a user can be classified in a category out of a set of categories $C_u = \{$ `moderate`, `spammer`, `pornographic`, `rude`, `violent`$\}$. For instance, a moderate is a user that mostly uploads correct contents, while a spammer tends to upload spam contents. Users may complain about contents. When complaining, users must label the complaint with a category out of a set of categories $C_{cm} = \{$`spam`, `porn`, `insult`, `violent`$\}$. Our simulator assumes that when a user complains about a content, the category of the complaint is the same category as the content. As an example, consider a user that views a `spam` content and finds it unacceptable. Then, she reports a complain which is considered to be of `spam` category.

Our simulator provides facilities to configure, execute and monitor simulations: (i) a scenario display that allows to visualise users' interactions in the on-line community; and (ii) an agents' population design tool that allows to design populations of agents with different profiles. Hence, in order to execute a simulation, a designer must first design a population by means of the population design tool, and subsequently run a simulation with the configured population. Additionally, our simulator allows to run single simulations that can be monitored with a graphical display, or, alternatively, to run simulations in background to perform experiments. The results of background executions can be analysed by means of charts. Furthermore, the simulator is prepared to execute simulations in background in a computer cluster.

In what follows we describe the scenario display, as well as the population design tool.

### 2.2. *The On-line Community Display*

Our simulator provides tools to display information about simulation runs. On the one hand, it provides a display to show qualitative information about the state of the scenario at each tick. On the other hand, it provides an inspector panel to show quantitative information about contents and their associated complaints. Figure 1 shows a screen shot of the display. Our simulator represents an on-line community as a grid where each cell corresponds to a position where a user can upload a content. Columns in the grid are grouped into the three different content sections previously introduced: `The Reporter`, `Forum`, and `Multimedia`. Each row corresponds to an agent in the user population of the on-line community. Whenever a user uploads a new content into a section, it is displayed into the next empty cell for that section in the corresponding user's row. Different types of contents may be distinguished to be `correct` (displayed in light-green in the figure with an "OK" label) from

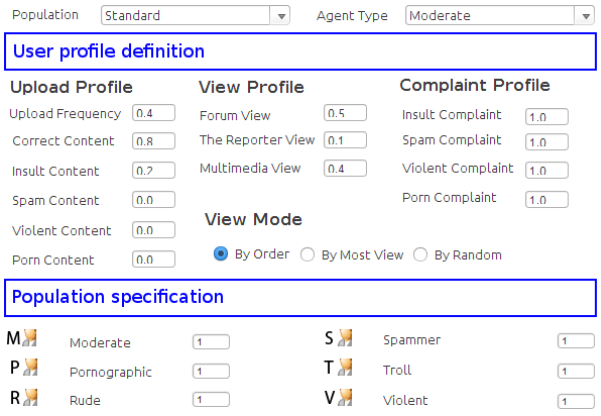



Fig. 2. On-line Communities Simulator: User profile definition and population specification.

(yellow/red coloured) conflicting ones, with the corresponding label in the display. When clicking on a content, the inspector panel shows its number of views. Contents with associated complaints are marked with a blue as exclamation mark below the content. When clicking on the complaints mark, the inspector panel shows the number of complaints that are associated to the corresponding content.

### *2.3. Designing Users' Populations*

Within an on-line community we may find users with different preferences and behaviours. With the aim of modelling different types of users, our simulator incorporates a population design tool, which is shown in Figure 2. It allows the designer to define users' preferences and describe their behaviours with regard to the contents they upload, view and complain about. The top of Figure 2 shows the current population that is being specified (depicted with label "Standard"). The bottom of the figure shows the composition of the population which is being designed. In particular, it shows a population of six different agents, which corresponds to the simulation run in Figure 1. For a given population, our design tool allows to create and define the profile of each user type (a moderate agent in the picture). A user profile describes how often a user uploads, views and complains about contents. Moreover, it also allows to specify the category of contents the user will choose to upload, view and complain about. As Figure 2 shows, a user profile contains three sub-profiles: the *upload profile*, the *view profile*, and the *complaint profile*. Next, we describe each sub-profile.

**The upload profile** describes: (i) the *upload frequency* of the user, namely the probability of the user of uploading contents at a given tick; and (ii) different *upload probabilities* for each content category. Thus, whenever a user uploads a content, the category of this content (correct, spam, porn, insult and violent) depends on some probability. Notice that all the probabilities assigned to different types of contents must sum up 1. As an example, the left part of Figure 2 shows the upload profile of a moderate user. It describes a user that uploads contents at each tick with probability 0.4. Every time she uploads contents, she has probability 0.8 of uploading correct contents, while she uploads insults with probability 0.2.

**The view profile** defines users' preferences in terms of the probability of viewing contents from each section of the community. The central part of Figure 2 shows a user's view profile. Likewise the upload profile, all the probabilities assigned to different sections must sum up 1. Moreover, the view profile also considers the *view mode*, which describes three different ways to choose contents to view:

1. *By order*. This method chooses the most recently uploaded contents in the community. Specifically, this method works as follows. First, it sorts all uploaded contents by the time they were uploaded. Second, it assigns a probability to each content according to a gamma distribution ($\Gamma(1,2)$).[1] Third, it randomly chooses the next content to view, according to the gamma distribution.
2. *Most viewed*. It chooses the contents with a larger number of visits. Specifically, it sorts contents by number of views, assigning to each content a probability according to a gamma distribution ($\Gamma(1,2)$).
3. *Randomly*. This method chooses contents according to a uniform random distribution.

**The complaint profile** defines the probability of a user of complaining about each type of visited content. Notice that, unlike previous probabilities, these values are independent and, hence, their addition is not required to be 1. The right hand side of Figure 2 depicts the complaint profile of a moderate user that complains about all type of conflicting contents with probability one.

Notice that, even though two users may have the same

[1] A graphical representation of this distribution can be found at `http://en.wikipedia.org/wiki/File:Gamma_distribution_pdf.svg`

| Upload Profile | | View Profile | | Complain Profile | |
|---|---|---|---|---|---|
| **Type** | *P* | **Section** | *P* | **Type** | *P* |
| Correct: | 1 | The Reporter: | 0.4 | Correct: | 0 |
| Spam: | 0 | Forum: | 0.4 | Spam: | 1 |
| Porn: | 0 | Multimedia: | 0.2 | Porn: | 1 |
| Insult: | 0 | TOTAL | 1 | Insult: | 1 |
| Violent: | 0 | ***View Mode*** | | Violent: | 1 |
| TOTAL: | 1 | By Order | ○ | | |
| ***Upload Frequency*** | | Most Viewed | • | | |
| Frequency: | 1 | Random | ○ | | |

Table 1
A *single-behaviour* moderate user's profile (*P* stands for probability)

category, their user profiles may be different. As an example, a moderate user may upload 100% correct contents, while another moderate user may upload 95% correct contents and 5% violent contents. As a convention, whenever a user is configured to upload a single content category with probability 1, we say that she is a *single-behaviour* user. As an example, table 1 depicts a single-behaviour moderate user's profile, since she has probability 1 to upload correct contents, while she has probability 0 to upload any other type of content. By contrast, whenever a user is configured to upload more than one content category with probabilities lower than 1, we say that she is a *mixed-behaviour* user. Figure 2 depicts a mixed-behaviour moderate user's profile, since she has probability 0.8 of uploading correct contents and probability 0.2 of uploading insults.

## 3. Background

At this point we have described our on-line communities simulator. We now survey IRON, an abstract and domain-independent online norm synthesis mechanism that we employ to regulate on-line communities. With this aim, IRON synthesises norms that are employed by the users of a community to avoid conflicts. Figure 3 illustrates the abstract architecture of IRON. In brief, it works by continuously monitoring agents' interactions in a distributed manner through its *sensors*, searching for conflicting situations. At each system's time step, IRON carries out the overall norm synthesis process throughout three subsequent stages. Firstly, whenever it detects new conflicts it performs a *norm generation* process that synthesises norms for the agents of the MAS. These new norms, which regulate agents' behaviour and are aimed at avoiding conflicts in the future, are then communicated to the agents of the MAS. Secondly, since at each time step agents choose whether to comply or not with norms, IRON monitors the consequences of such decisions to evaluate norms. In other words, it performs a *norm evaluation* process to assess if norms manage to avoid conflicts or not. Thirdly, it carries out a *norm refinement* process, which: (i) generalises norms when possible, joining several norms to a single parent that concisely represents all of them; and (ii) discards those norms that have not performed well for a period of time. Fourthly, if IRON has made any changes to the normative system, either by adding or removing norms, it sends the new normative system to the agents within the MAS. Notice then that IRON's norm synthesis is a *conflict-driven* process. It generates norms whenever new conflicts arise, and it evaluates norms based on the conflicts that arise after agents fulfil or infringe norms. Moreover, it generalises and specialises (deactivates) norms based on their continuous evaluations. Therefore, conflicts also affect norm refinement.

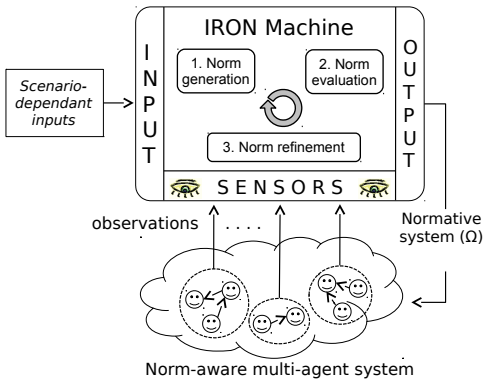



Fig. 3. IRON's abstract architecture.

Before detailing IRON's norm synthesis process, first we will detail IRON's information model. Consider a Multi-Agent System (MAS) composed of a set of agents $Ag = \{ag_1, \ldots, ag_n\}$ and a set of actions $Ac = \{ac_1, \ldots, ac_c\}$ available to the agents. Agents in the system have their local, *individual context*, which is described in terms of their local point of view (that is, mimicking their perception) In IRON, norms are of the form $\langle \varphi, \theta(ac) \rangle$ where $\varphi$ is the norm's precondition, $\theta$ is a deontic operator (e.g., a *prohibition*) and $ac \in Ac$ is an action available to the agents. The precondition of a norm is a set of first-order n-ary predicates $p(\tau_1, \ldots, \tau_n)$, where $p$ is a predicate symbol and $(\tau_1, \ldots, \tau_n)$ is a set of terms. In IRON, norms are expressed in terms of agents' individual contexts so that they can be understood by them. Hence, whenever the individual context of an agent satisfies the precondition $\varphi$ of a norm, then the norm applies to the agent and $\theta(ac)$ holds for it. We define a normative system

$\Omega$ as the set of norms that are currently active in the multi-agent system.

As an example, consider a traffic scenario where agents are driving cars, and conflicting situations are *collisions* between cars. We consider three unary predicate symbols $\{left, front, right\}$ representing the three road positions that an agent perceives. Each predicate has a single term from $\{police, ambulance, fire-brigade, emergency, nil\}$ representing different types of *emergency* vehicles, and symbol "*nil*" standing for no vehicle. The actions available to agents are $Ac = \{go, stop\}$. With these definitions in place we can create norms such as $n_1, n_2, n_3$. All three norms establish a *prohibition* to *go* to an agent that has different types of emergency vehicles to its left position, and nothing to its front and right positions.

$n_1 : \langle \{left(police), \quad front(nil), right(nil)\}, prh(go) \rangle$
$n_2 : \langle \{left(ambulance), front(nil), right(nil)\}, prh(go) \rangle$
$n_3 : \langle \{left(fire\text{-}brigade), front(nil), right(nil)\}, prh(go) \rangle$

IRON is an abstract, domain independent mechanism. However, in order to configure it for different scenarios, it requires some domain-dependant inputs that must be implemented for each specific scenario. In what follows, we will describe IRON's norm synthesis process as well as the inputs it requires to synthesise norms for a specific scenario.

### 3.1. Norm Synthesis Process

IRON is based on three components to perform norm synthesis: (i) a *normative network* (NN), which is a graph-based data structure to represent explored norms, (ii) a set of normative network *operators* that allow to apply changes to the normative network, and (iii) a *strategy* to apply operators to the normative network. The right-hand side of Figure 4 illustrates the components of IRON. The *control unit* contains the set of operators $O$, as well as a strategy $\Pi$ to apply operators. IRON's strategy applies operators to the normative network in order to retrieve information about norms and to apply changes to the normative network (see *read* and *write* arrows in Figure 4). In what follows, we briefly describe these three main components.

#### 3.1.1. The Normative Network

IRON represents explored norms by means of the normative network (NN). Specifically, a normative network is a graph-based data structure whose nodes stand for norms and whose edges stand for (generalisation)

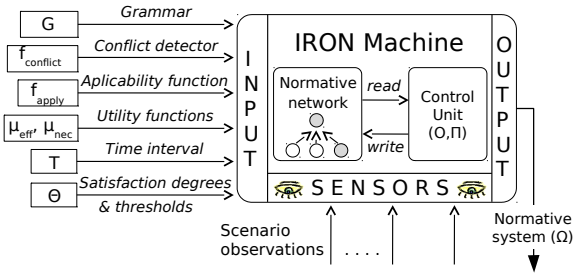



Fig. 4. IRON's components and inputs.

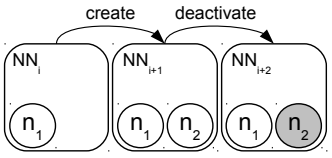



Fig. 5. Evolution of a Normative Network along time.

relationships among norms. In a normative network, norms can be either *active* or *inactive*. The set of active norms in the normative network constitutes the *normative system* that is provided to the agents in the scenario. Figure 5 illustrates the evolution of a normative network (and its corresponding normative system) over time period $t_i - t_{i+2}$. At time step $t_i$, the normative network $NN_i$ contains one unique active norm $n_1$ (represented as a white circle), hence representing the normative system $\Omega_i = \{n_1\}$. At time $t_i$ a new active norm $n_2$ is created and added to the normative network, hence leading to $NN_{i+1} = \{n_1, n_2\}$ and $\Omega_{i+1} = \{n_1, n_2\}$. Finally, at instant $t_{i+2}$ norm $n_2$ is deactivated (represented as a gray circle), yielding $NN_{i+2} = \{n_1, n_2\}$ and $\Omega_{i+2} = \{n_1\}$. Notice that at time step $t_{i+2}$ the normative system contains one unique norm $n_1$, even though the normative network contains two norms $n_1, n_2$. Recall that a normative network represents a normative system as its active norms, and at time $t_{i+2}$ the only active norm in the normative network is $n_1$.

#### 3.1.2. Operators for normative networks

IRON transforms the normative network over time, leading from one normative system into another, searching for a normative system that effectively coordinates the MAS. With this aim, it includes four different operators. More precisely, IRON implements operators to perform: (i) the *creation* of a new norm, activating it and adding it then to the normative network, (ii) the *deactivation* of a norm in the normative network, hence removing it from the normative system, (iii) the *generalisation* of a group of norms in the normative network to a more general norm that concisely represents

all of them, and (iv) as a dual operation to generalisation, the *specialisation* of a general norm to more specific norms. Next we detail each operator:

**Add**. The *add* operator adds a norm to the normative network so that IRON can keep track of its ability to avoid conflicts.

**Activate**. As described above, norms in a normative network may be either *active* or *inactive*. Operator *activate* sets the state of a norm in the normative network to *active* so that it belongs to the normative system.

**Create**. The *create* operator synthesises a new norm from each new detected *conflict*. Next, it employs operators *add* and *activate* to activate the norm and add it to the normative network. Thus, the created norm will be included in the normative system. IRON assumes that a conflict can be avoided if some of the agents' previous actions are not performed. Thus, it generates norms that prohibit agents to perform such actions in the same conflictive context. Specifically, the *create* operator receives a set of detected $conflicts$ and generates a new norm for each conflict. This generation process is based on an unsupervised version of classical Case-Based Reasoning (CBR) [1] (more details can be found at [12]). Figure 5 illustrates the creation of a norm. At time $t_i$, the normative network $NN_i$ contains one active norm $n_1$. By applying operator *create*, a new active norm $n_2$ is added to the normative network, yielding $NN_{i+1} = \{n_1, n_2\}$ and $\Omega_{i+1} = \{n_1, n_2\}$.

**Deactivate**. Consider that at a given time IRON detects that a norm does not succeed in avoiding conflicts, and hence must be removed from the normative system. IRON will not remove the norm from the normative network, but it will use operator $deactivate$ to set the state of the norm to *inactive*. Therefore, the norm will no longer belong to the normative system but the normative network will still have it to keep track of its exploration. Again, Figure 5 depicts the normative network $NN_{i+1}$, which at time $t_{i+1}$ contains two active norms $n_1, n_2$. Then, operator $deactivate$ deactivates norm $n_2$, yielding $NN_{i+2} = \{n_1, n_2\}$ and $\Omega_{i+2} = \{n_1\}$.

**Generalise**. As part of the norm refinement process, IRON uses operator generalise to represent several norms as a more general, single norm that implicitly includes all of them. Specifically, norm generalisations can be performed for any norm in the normative network. Thus, by means of norm generalisations IRON reduces the cardinality of synthesised normative systems. As an example, consider norms $n_1, n_2, n_3$ described above in this section. All these three norms can be generalised to the following norm:

$$n_4 : \langle \{left(emergency), front(nil), right(nil)\}, prh(go) \rangle$$

Norm $n_4$ prohibits an agent to go whenever it perceives any type of *emergency* vehicle (either police, ambulance or fire-brigade) to its left position, and nothing to its front and right positions. Figure 6 illustrates the generalisation of $n_1, n_2, n_3$ to $n_4$. At time $t_i$, the normative network $NN_i$ contains three active norms $n_1, n_2, n_3$. Then, IRON generalises them to $n_4$ by performing the following steps: (i) it generates norm $n_4$, (ii) it activates $n_4$ and adds it to the normative network, (iii) it establishes generalisation relationships from $n_1, n_2, n_3$ to $n_4$, and (iv) it deactivates norms $n_1, n_2, n_3$, removing them from the normative system.

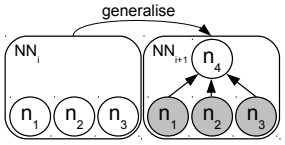



Fig. 6. Generalisation of norms $n_1, n_2, n_3$ to a new norm $n_4$.

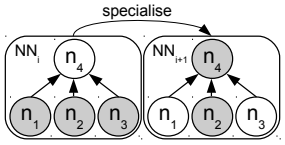



Fig. 7. Specialisation of norm $n_4$ to its child norms $n_1, n_3$.

**Specialise**. Operator *specialise* undoes a norm generalisation, specialising a general norm into a set of more specific norms. As an example, consider the normative network $NN_i$ depicted in Figure 7. It represents normative system $\Omega_i = \{n_4\}$, containing one single norm which is applicable in the specific situations described by norms $n_1, n_2, n_3$. Consider now norm $n_2$ does not succeed in avoiding conflicts, and hence it must be removed from the normative system. Even though $n_2$ does not belong to the normative system, it is implicitly represented by $n_4$. Therefore, operator *specialise* specialises $n_4$, deactivating it and activating $n_1, n_3$ in the normative network. Thus, after the norm specialisation, the normative system becomes $NN_{i+1} = \{n_1, n_3\}$, which no longer contains the situation described by $n_2$.

### *3.1.3.* IRON*'s strategy*

IRON invokes previous operators by following a specific *strategy* to perform the norm synthesis process. Specifically, the proposal of [12] is to monitor the evolution of the system at regular time intervals and apply operators under certain conditions. The strategy that IRON follows may be divided into three consecutive stages: *norm generation, norm evaluation* and *norm refinement*. Next we detail each stage of the norm synthesis strategy.

**Norm generation**. During this phase, IRON first monitors the multi-agent system operation through a set of distributed *sensors*, searching for conflicts. As depicted in Figure 4, IRON represents agents' perceived interactions in the form of *observations*, which are partial descriptions of the scenario from a global, external observer's perspective. Then, it performs conflict detection within perceived *observations*. Whenever conflicts arise, it invokes the previously described operator *create* to generate norms that regulate agents' behaviour in order to avoid detected conflicts in the future. Recall that, since agents must be able to understand norms, IRON describes norms from an agent's local perspective.

**Norm evaluation**. At each time step, some norms may apply to the agents. In this case, agents decide whether to fulfil norms or infringe them. During the norm evaluation stage, IRON monitors the effects of such decisions in order to assess if norms succeed in avoiding conflicts. With this aim, it determines if agents have fulfilled or infringed norms, and which of these fulfilments or infringements have led to conflicts. Then, norms are evaluated in terms of their *effectiveness* and *necessity*, represented as $\mu_{eff}$ and $\mu_{nec}$. On the one hand, IRON measures the effectiveness $\mu_{eff}$ of a norm from the outcomes of its *fulfilments*: the higher the ratio of *successful* fulfilments (fulfilments that did not end up with conflicts), the more effective the norm. On the other hand, it measures the necessity $\mu_{nec}$ of a norm according to the following principle: the higher the ratio of *harmful infringements* (infringements leading to conflicts), the more necessary the norm. Finally, IRON computes the effectiveness and necessity *ranges* of norms during a period of time $T$. Specifically, a range of a norm contains a lower and upper bound for the range of values of effectiveness or necessity $(\mu_{eff}, \mu_{nec})$ of the norm during $T$. These ranges will be used, as we show below, to refine norms.

**Norm refinement**. The norm refinement process yields a new normative system transforming the normative network, deactivating ineffective or unnecessary norms, and performing norm generalisations and specialisations. On the one hand, it generalises a norm whenever the lower bound of its effectiveness *and* necessity ranges are over a generalisation threshold $\alpha_{gen}$. On the other hand it specialises (deactivates) a norm whenever the upper bound of its effectiveness *or* necessity ranges are under a specialisation threshold $\alpha_{spec}$.

### *3.2. IRON's inputs*

As detailed above, IRON's norm synthesis is an abstract, domain-independent mechanism. However, during the norm synthesis process it requires some domain information: (i) a grammar $\mathcal{G}$ to define norms for the given scenario; (ii) a conflict detection function $f_{conflict}$ that allows to detect conflicts in perceived *observations*; (iii) a norm applicability function $f_{apply}$ to determine whether a norm applies to the agents in a perceived *observation*; (iv) norm evaluation functions to compute the effectiveness and necessity of norms in the normative network; and (v) a set of configuration parameters composed of a time interval ($T$) to compute effectiveness and necessity ranges of norms during a period of time $T$, as well as a set of thresholds to define the acceptable range of effectiveness and necessity of norms, to consider to generalise and specialise norms.

#### *3.2.1. A Grammar for norm synthesis*

The first input IRON requires is a grammar to synthesise norms of the form $\langle \varphi, \theta(Ac) \rangle$ for the given scenario. IRON adapts its grammar from [4], using as building blocks *atomic formulae* of the form $p^n(\tau_1, \ldots, \tau_n)$, $p$ being an n-ary predicate symbol and $\tau_1, \ldots, \tau_n$ terms of an agents' language that describes agents' individual contexts.

$$
\begin{array}{lll}
\text{Norm} & ::= & \langle \varphi, \theta(Ac) \rangle \\
\varphi & ::= & \varphi \,\&\, \varphi \mid \alpha \\
\theta & ::= & obl \mid prh \\
Ac & ::= & ac_1 \mid ac_2 \mid \ldots \mid ac_n \\
\alpha & ::= & p^n(\tau_1, \ldots, \tau_n)
\end{array}
$$

In order to configure IRON to synthesise norms for a given scenario, this abstract grammar must be instantiated, specifying the predicates and terms that are considered for that particular scenario.

#### *3.2.2. A function for conflict detection*

The definition of conflict is domain-dependant. For instance, in a traffic scenario a conflicting situation may be a collision among cars, while in an on-line community scenario a conflict may be defined as a user that uploads inappropriate contents (e.g., uploading a `spam` content), hence leading other users to complain about it. Therefore, IRON requires as an input a function $f_{conflict}$ to detect conflicts for a given scenario. Specifically, function $f_{conflict}$ receives as an input a set of perceived *observations*, and returns a set of *conflicts* that it has detected within these *observations*.

*3.2.3. A function to detect norm applicability*

As described in Section 3.1.3, IRON evaluates norms based on the consequences of agents' norm fulfilments and infringements. With this aim, function $f_{apply}$ receives as an input a set of perceived *observations*, and returns a set of norms that apply in the situations described by the *observations*. Specifically, this function operates as follows. First, it computes the individual context of each agent in the *observation*. Second, for each agent's individual context, it retrieves which norms apply to that context. As an example, consider an agent which perceives a police car to its left, and nothing to its front and right positions. In this specific individual context, norm $n_1$ applies to the agent and hence the agent is forbidden to go.

*3.2.4. Functions to evaluate norms*

During norm evaluation, IRON requires as an input two functions $\mu_{eff}$ and $\mu_{nec}$ to evaluate the effectiveness and necessity of norms. In fact, these functions are not really scenario-dependant. They evaluate norms based on the conflicts (in general) that arise after agents fulfil or infringe norms. However, IRON provides two default utility functions $\mu_{eff}$ and $\mu_{nec}$ that may be replaced by other functions explicitly implemented for a specific scenario. IRON's default utility functions evaluate norms along the lines of the explanation of norm evaluation in section 3.1.3. Function $\mu_{eff}$ computes the effectiveness of a norm as its ratio of successful fulfilments. Function $\mu_{nec}$ computes the necessity of a norm as its ratio of harmful infringements. The definition of default formulas $\mu_{eff}, \mu_{nec}$ correspond to the formulas 1, 2, 3, 4 in [12].

*3.2.5. Configuration parameters*

Recall from section 3.1.3 that IRON refines the normative system deactivating, generalising and specialising norms, based on their effectiveness and necessity ranges and a set of generalisation and specialisation thresholds. Therefore, IRON requires as an input parameter $T$ to compute ranges, as well as thresholds $\alpha_{gen}, \alpha_{spec}$ to generalise and specialise norms. A low time interval $T$, will make IRON to be more *reactive* to changes in effectiveness and necessity ranges, refining the normative system in consequence. By contrast, large time intervals $T$ will make IRON to be more *conservative* to refine the normative system, since effectiveness and necessity ranges will be less reactive to changes. Moreover, the greater the generalisation threshold is, the more conservative IRON is about generalising norms. Finally, the lower the specialisation threshold is, the more conservative IRON will be about deactivating norms.

## 4. Frictionless On-line Communities: Exploiting IRON to Synthesise Norms

At this point we have described our on-line communities simulator, as well as IRON, a mechanism for the on-line automated synthesis of norms for MASs. We now employ IRON to build a participatory legislation mechanism for an on-line community, where community users participate in the regulation process. With this aim, in this section we describe how to configure IRON to synthesise norms for the agents in our on-line communities simulator.

Even though IRON is an abstract mechanism (i.e., scenario independent), during the norm synthesis process it employs some elements that are scenario-dependant. First, IRON's perceived *observations* are scenario-dependant, since each scenario may be composed of different elements, and may be described in different ways. Second, in order to generate, evaluate and refine norms, IRON requires some scenario-dependant inputs (described in Section 3.2), which allow to generate norms for the scenario, as well as detecting conflicts, norm applicability and evaluate norms. In what follows we provide the specification of IRON's perceived *observations*, as well as IRON's scenario-dependant inputs.

### *4.1. Perceiving On-line Communities*

Recall from Section 2 that our on-line community scenario is divided into three different sections where the users upload, view and complain about contents. In order to perceive the scenario, at each time step the simulator generates *observations* for IRON. A *observation* is composed of three lists, one for each section. Each element in a list contains a content that has been uploaded, viewed, or complained about, as well as the corresponding action, and the identifier of the user who performed the action. Moreover, each content incorporates the information about its category and its owner, namely the user that uploaded the content. As an example, consider that during the current time step user $u_1$ has uploaded a `correct` content in section `Forum`, and user $u_2$ has viewed and complained about a `spam` content (whose owner is user $u_1$) in section `Multimedia`. The corresponding observation can be thus described as shown in Table 2. In section `Forum`, it shows user $u_1$ that has performed the action `upload` over a content of category `correct`. In section `Multimedia`, the table shows that user $u_2$ has performed two actions: a `view` and a `complain` over a content with category `spam`, whose owner is user $u_1$.

$$n : \langle (user(u_1), \quad section(\texttt{Multimedia}), \quad contentCatg(\texttt{spam})), \quad prh(\texttt{upload}) \rangle$$
$$n' : \langle (user(u_3), \quad section(\texttt{Forum}), \quad contentCatg(\texttt{violent})), \quad prh(\texttt{upload}) \rangle$$

Fig. 8. Examples of norms for the on-line community scenario.

| The Reporter | Forum | Multimedia |
|---|---|---|
| | content: $id(1)$<br>• action: $u_1 \rightarrow$ upload<br>• category: correct<br>• owner: $u_1$ | content: $id(2)$<br>• action$_1$: $u_2 \rightarrow$ view<br>• action$_2$: $u_2 \rightarrow$ complain<br>• category: spam<br>• owner: $u_1$ |

Table 2
An example of an IRON's *observation*

### 4.2. Norms for On-line Communities

We must provide IRON with a specific grammar to synthesise norms for the on-line community scenario. By following the grammar specification in Section 3.2.1, we instantiate our grammar as follows. On the one hand, a norm precondition has three unary predicates with predicate symbols in $P = \{user, section, contentCatg\}$. The term for predicate *user* is one out of the set of user identifiers $U$. The term for predicate *section* is one out of the set of section names $S = \{\texttt{Forum}, \texttt{Multimedia}, \texttt{The Reporter}\}$. The term for predicate *contentCatg* is one out of the set of content categories $C_{ct} = \{\texttt{correct}, \texttt{spam}, \texttt{porn}, \texttt{insult}, \texttt{violent}\}$. On the other hand, we consider each norm's consequence only specifies a prohibition to perform the action upload of the content in the context (thus, the deontic operator is $\theta = phr$). Therefore, norms establish prohibitions for users to upload certain types of contents to some community sections. Figure 8 shows two examples of norms $n$ and $n'$ automatically synthesised by IRON by means of this grammar. Norm $n$ prohibits user $u_1$ to upload spam contents into the Multimedia section, while norm $n'$ prohibits user $u_3$ to upload violent contents into the Forum section. Recall from Section 3 that IRON generates new norms whenever it detects new conflicts in the scenario. In what follows we explain how to detect conflicts in our on-line community scenario.

### 4.3. Detecting Conflicts in On-line Communities: A Participatory Approach

Function $f_{conflict}$ identifies conflicts in a *observation* perceived by IRON. Our definition of $f_{conflict}$ is aimed at letting users jointly decide what type of situations represent a conflict for them (e.g., the upload of an offensive content). Particularly, in our on-line community scenario, conflicts can be identified based on the complaints users report about contents. Consider that a user uploads some content, and then that content receives a number of complaints which is high enough with respect to the number of views. In that case, the action of uploading the content is considered as a conflict. Recall from Section 3 that IRON's norm synthesis is conflict-driven, since it generates, evaluates and refines norms based on conflicts. Therefore, thanks to our definition of conflict, the whole norm synthesis process becomes a participatory process which is guided by users' complaints.

Our instantiation of function $f_{conflict}$ is designed to assess if the contents within a given *observation* are conflictive or not. In particular, for each given content it uses function $conflictive(content)$ to check if it is conflictive. This function computes the *complaint ratio* of contents that have been viewed at current time step. The complaint ratio of a content is computed as the ratio of the accumulated number of complaints over its total number of views. This function returns true if the content: (i) has a minimum number of $N$ views (to ensure that the conflict computation is performed with enough evidence); and (ii) is considered to be conflictive, namely if its *complaint ratio* is larger than a conflict threshold $\alpha_{conflict} \in [0,1]$:

$$conflictive(content) = \begin{cases} true & \textbf{if } views(content) \geq N \textbf{ and} \\ & \frac{complaints(content)}{views(content)} > \alpha_{conflict} \\ false & \textbf{otherwise} \end{cases}$$

As an example, consider that user $u_1$ uploads a spam content in section Multimedia. Consider now that other users view the content and complain about it, reporting a complaint of type spam. In case the *ratio* between users' complaints about that content and its total number of views is over threshold $\alpha_{conflict}$, then uploading spam contents in section Multimedia is considered as a new conflict. Therefore, the users of the on-line community trigger, by means of their complaints, the generation of norm $n$ (depicted in Figure 8), which prohibits user $u_1$ to upload spam contents into section Multimedia.

### 4.4. Detecting norm applicability

Our implementation of the applicability function $f_{apply}$ required by IRON works as follows. Given a

*observation*, it performs the following steps: (i) it generates a list with the identifiers of the users that have uploaded new contents into the community during the current time step; (ii) it computes the individual context of each user that has uploaded contents; and (iii) it retrieves the norms that apply to each agent's individual context. As an example, consider that the current normative system is $\Omega = \{n, n'\}$. Consider now that IRON perceives that user $u_1$ uploads a spam content in section `Multimedia`. First, function $f_{apply}$ builds a list with the identifier of user $u_1$, since it has recently uploaded contents. Second, it computes $u_1$'s individual context in the same format than the precondition of a norm as specified by the grammar. Thus, $u_1$'s individual context can be compared with the preconditions of norms in order to detect what norms apply to the user. For instance, say that $u_1$'s individual context is:

$$(user(u_1), section(\texttt{Multimedia}), contentCatg(\texttt{spam}))$$

Third, function $f_{apply}$ retrieves the norms within the current normative system that apply to this specific individual context. Notice that the individual context of $u_1$ is equal to the precondition of norm $n$. Therefore, norm $n$ applies to user $u_1$, and hence the prohibition to upload spam contents to section `Multimedia` holds for her.

### *4.5. Evaluating norms in on-line Communities*

Recall from Section 3.2.4 that IRON provides default functions to evaluate norms in terms of their effectiveness and necessity. Within our on-line community scenario, norms cannot be evaluated in terms of their effectiveness since norm applicability is not observable. As an explanation, recall that (i) norms are aimed at prohibiting users to upload conflicting contents, and (ii) the effectiveness of a norm is computed from the outcomes of its *fulfilments*. Whenever a user fulfils a norm, then it does not upload conflicting contents and conflicts do not arise. As a consequence, IRON cannot detect whether the absence of conflicts is because either the user fulfilled a norm or because she did not have the intention to upload conflicting contents.

By contrast, norms can be evaluated in terms of their necessity. Whenever a user uploads conflicting contents, then conflicts arise and IRON can detect that the user has infringed a norm. Specifically, we evaluate norms' necessity by means of IRON's default necessity function $\mu_{nec}$. Specifically, the necessity of a norm is computed as its ratio of harmful infringements. On the one hand, norm infringements that lead to conflicts make IRON to evaluate the norm as necessary. On the other hand, norm infringements that lead to non-conflictive situations make IRON to evaluate the norm as unnecessary.

## 5. Empirical Evaluation

We now perform a preliminary empirical evaluation of IRON's norm synthesis for the online community scenario described in Section 2. We first detail in section 5.1 the empirical settings of our experiments. Thereafter, in section 5.2 we empirically show that IRON is capable of synthesising norms for on-line communities, searching for a normative system that avoids conflicts, hence satisfying the community users by avoiding frictions.

### *5.1. Empirical settings*

Our experiments employ the simulator described in Section 2 to simulate an on-line community scenario. As depicted in table 3, in addition to the user population design tool, the simulator provides a set of parameters to configure simulations. The table also shows IRON's parameters, which allow to configure the norm synthesis process performed by IRON. As initial experiments with our simulator, in our experiments we consider different populations composed of 10 agents that model community users. In particular, our populations are composed of single-behaviour moderate users and spammers. On the one hand, single-behaviour moderates upload correct contents with probability 1. Moderate users complain about spam contents according to a probability (*complain rate* or *CR* hereafter), which is defined in their individual complain profiles. On the other hand, single-behaviour spammers upload spam contents with probability 1 (divided equally into the three sections of the community), while they never complain about spam contents.

We aim at studying how IRON manages to converge to a stable normative system that avoids conflicts when the users' population is composed of different percentages of complaining users (i.e., moderate users). With this aim, we perform our empirical evaluation for:

1. A population with a *majority* of complaining users, composed of 7 moderates and 3 spammers.
2. A *balanced* population, composed of 5 moderates and 5 spammers.
3. A population with a *minority* of complaining users, composed of 3 moderates and 7 spammers.

| Parameter | Description | Value |
|---|---|---|
| *CR* | Users probability to complain about inappropriate contents | See Section 5.1 |
| *NIR* | Norm infringement rate | 30%,50%,70% |
| *N* | Min. number of views to assess if a content is conflicting | 10 |
| $\alpha_{conflict}$ | Threshold to evaluate contents as conflictive | 0.3 |
| *T* | Window size to compute norms performances | 100 |
| $\alpha_{gen}$ | Norm generalisation threshold | 0.6 |
| $\alpha_{spec}$ | Norm specialisation threshold | 0.2 |
| $w_{I_C}$ | Importance of harmful infringements. | 1 |
| $w_{I_{\bar{C}}}$ | Importance of harmless infringements. | 2 |

Table 3
Empirical evaluation parameters

At each tick, each user decides whether to comply or not with the norms published by IRON according to some probability, namely a *norm infringement rate*. The probability of violating norms is a parameter NIR which is fixed for each simulation and is the same for all users. Consider a user with profile probability 0.5 of uploading contents. Consider now that a norm prohibiting to upload content holds for this user, which in turn has probability 0.3 of violating norms. First, the user uses its profile probability of uploading contents (which is 0.5) to decide whether updating a content or not. In case she decides to upload a content, then she uses the norm infringement rate (0.3) to decide if she still wants to upload the content (thus, infringing the norm) or if she aborts its initial uploading purpose so that she fulfils the norm.

Each simulation finishes when it reaches 3,000 ticks. We consider a "*warm-up*" period of 500 ticks for all simulations: from tick 0 to 500, users only upload contents, and from tick 500 onwards, users upload contents and also view and complain about contents. The *warm-up* period allows a simulation to reach normal conditions in on-line communities, where users enter in the community and there already exist contents to view and complain about. Contents are considered to be conflictive whenever they: (i) have a minimum number of $N = 10$ views; and (ii) their *complaint ratio* is over a threshold $\alpha_{conflict} = 0.5$

Regarding IRON's configuration parameters (described in section 3.2.5), we have taken a conservative approach to set them. That is intended to refine the normative system only when it is necessary. In our particular scenario, we have established a time interval ($T = 100$) that leads IRON to compute necessity ranges with a large number of values along time. Regarding IRON's thresholds, we just generalise norms that perform well (with high necessity), and to deactivate norms that perform poorly. Thus, the generalisation threshold has been set to a high value ($\alpha_{gen} = 0.6$), and the specialisation threshold has been set to a low value ($\alpha_{spec} = 0.2$). As for the IRON's function $\mu_{nec}$ to evaluate norms' necessity, it requires to set two weights $w_{I_C} > 0$, $w_{I_{\bar{C}}} > 0$ that measure the importance of harmful infringements and harmless infringements of a norm, respectively. In our experiments, we have decided to set weights as they were set in the empirical evaluation in [12]. Specifically, the chosen values are $w_{I_C} = 1$, $w_{I_{\bar{C}}} = 2$. This means that a norm infringement that did not end up with conflicts is two times more important than a norm infringement that led to conflicts.

Finally, we consider that IRON converges to a normative system if during a 1,000-tick period: *(i)* no new conflicts arise; *(ii)* the normative system contains norms; and *(iii)* the normative system remains unchanged.

### 5.2. Empirical results

We now analyse the results of our empirical evaluation. First, we perform a micro analysis of IRON's convergence process. Our purpose is to shed light on how IRON manages to synthesise norms from scratch based on users' complaints, yielding a normative system that avoids conflicting situations. Thereafter, we perform a macro analysis of IRON's percentage of convergence for different populations, namely its capability to synthesise normative systems that avoid conflicts.

#### 5.2.1. Micro analysis: IRON's convergence process

Our first analysis focuses on how IRON manages to synthesise norms from scratch for a given population, converging to a normative system that avoids conflicts. Specifically, we focus on the convergence process for a population with a majority of moderate users, since it is the one which approximates better the actual-world conditions of on-line communities. Therefore, we employ a population of 7 moderate users with 50% complain rate, and 3 spammers with 50% norm infringement rate. Figure 9 aggregates the results of 100 different simulations. It shows the cardinality of the normative system along time, the average of conflicts and the cardinality of the normative network. During the *warm-up* period (up to tick 500) users only upload contents, and therefore there are no user complaints. At tick 500, moderate users start complaining about spam contents that they view. These complaints lead IRON to consider the uploading of these contents as new con-

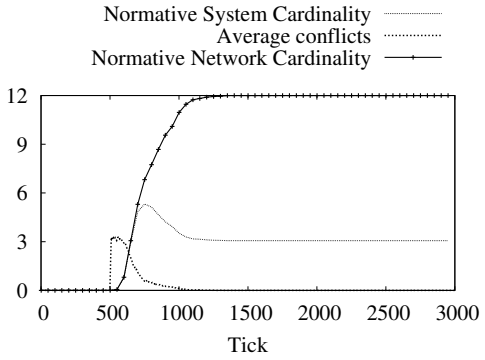



Fig. 9. IRON 's convergence process for a population of 7 moderate users and 3 spammers with complain rate 50% and 70% norm infringement rate.

$$n_a : \langle (user(u_8),\ contentCatg(\texttt{spam})),\ prh(\texttt{upload}) \rangle$$
$$n_b : \langle (user(u_9),\ contentCatg(\texttt{spam})),\ prh(\texttt{upload}) \rangle$$
$$n_c : \langle (user(u_{10}),\ contentCatg(\texttt{spam})),\ prh(\texttt{upload}) \rangle$$

Fig. 11. Norms that IRON synthesises to regulate spammers' behaviour.

flicts. Therefore, after tick 550, IRON generates new norms to prevent spammers from uploading spam contents in the different sections of the community. As a consequence, the cardinality of the normative network increases, as well as the cardinality of the normative system. Near tick 750, IRON performs norm generalisations, joining several norms into more general norms, decreasing the cardinality of the normative system. Near tick 1,000, IRON manages to synthesise a compact normative system of 3 general norms that regulate the on-line community. Notice that, by contrast, the normative network contains 12 norms. It contains: (i) nine norms to prohibit the three spammers to upload spam into the three sections of the community (3 spammers $\times$ 3 sections = 9 norms); and (ii) three general norms that IRON has synthesised to yield a compact normative system of 3 norms. Figure 11 shows the three norms $n_a, n_b, n_c$ that IRON converged to. Norms $n_a, n_b, n_c$ are *general* norms that prohibit users $u_8, u_9$ and $u_{10}$ to upload spam contents to any section of the on-line community. After the synthesis of these tree norms, the simulation converges and no new conflicts are detected from tick 1,100 onwards.

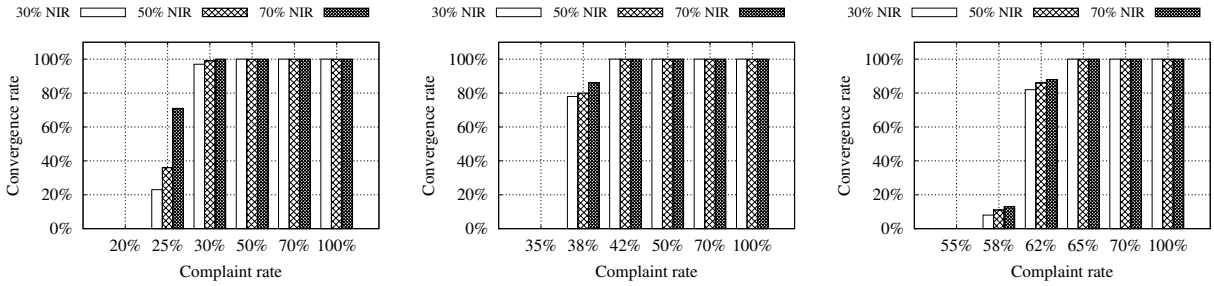
*5.2.2. Macro analysis:* IRON*'s convergence outcome*

We now analyse the convergence outcome of IRON for different populations of users. First, for each population we analyse IRON's convergence rate for different users. Furthermore, for each population we analyse its convergence for different users' behaviours regarding conflicting contents and norms. Second, we measure the compactness of the normative systems synthesised by IRON. This measure tells us the number of norms required to regulate all the conflicts detected during simulations. We compute compactness of a normative system $\Omega$ as $(1 - \frac{|\Omega|}{|NN|}) \times 100$, where $|\Omega|$ stands for the number of norms in the normative systems and $|NN|$ stands for the number of norms in the normative network (i.e., the total number of norms that IRON synthesised during simulations).

With this aim, we run simulations combining different: (i) *populations* with different percentages of complaining users (majority of complaining users, balanced population and minority of complaining users); (ii) *complain rates*, namely different probabilities for moderate users to complain about conflicting (spam) contents; and (iii) *norm infringement rates*, that is, different probabilities for spammers to violate norms that apply to them. Specifically, for each population we performed simulations for complain rates ranging from 0% to 100%, and for norm infringement rates ranging from 30% to 70%. We performed 100 different simulations for each combination of population, complain rate and norm violation rate. In what follows we show the empirical results, which are organised into different populations.

**1. Majority of complaining users.** We now analyse IRON's convergence rate with a population of 7 moderates and 3 spammers. Figure 10a illustrates IRON's convergence rate for this population, given different combinations of complain rates and norm infringement rates (depicted as NIR). For *very low* complaint rates (up to 20%), IRON never synthesises norms to regulate spammers' behaviour. Since moderate users rarely complain about spam contents, then these contents are never considered to be conflicting[2] and hence IRON, never triggers norm generation. Therefore, all simulations finish with an empty normative system. As the complaint rate increases, IRON's convergence rate increases as well. For *low* complaint rates, (between 25% and 30%), IRON enters into a *transition area* where the convergence rate starts to increase. Notice that, for a given complain rate, the convergence rate increases as long as the norm infringement rate increases. As an example, we now focus on the 25% complaint rate. For low norm infringement rates (30% of norm in-

[2]Recall from Section 4.3 that a content is considered to be conflictive whenever its ratio of complaints with respect to the total number of views is over a certain threshold.




a) Population: 7 moderates, 3 spammers. b) Population: 5 moderates, 5 spammers. c) Population: 3 moderates, 7 spammers.

Fig. 10. Convergence rate of IRON for different populations of users, users' complain rates and norm infringement rates (NIR).

fringements), IRON converges to a normative system with norms for 23% of the total simulations, while it converges 36% and 71% of the times for 50% and 70% norm infringement rates respectively. As the norm infringement rate increases, spammers have a higher probability to violate norms, hence uploading spam contents. These spam contents cause moderate users' complaints, leading IRON to detect new conflicts and to generate new norms. Finally, for *medium* and *high* complaint rates (from 50% on) IRON converges to a normative system 100% of the times. Here IRON benefits from the high number of complaints that moderate users report about spam contents, which trigger the generation of norms to regulate spammers' behaviour. For each simulation that converged, IRON found 9 conflicts, and hence synthesised 9 norms to regulate them, one per conflict. Thereafter, IRON managed to generalise those 9 norms to 3 new general norms. Overall, IRON synthesised 12 norms and managed to end up with a 3-norm normative system. Therefore, IRON converged to normative systems with 75% compactness.

**2. Balanced population of users.** Figure 10b illustrates the convergence rate of IRON for a population of 5 moderates and 5 spammers for different combinations of complain rates and norm infringement rates. In contrast with the previous population, for this population IRON starts generating norms when users' complaint rate goes beyond 38%[3]. Since this population has a lower percentage of complaining users than the previous population, then there exists a lower number of complaints, and IRON detects a lower number of conflicts. As a consequence, IRON starts to generate norms to regulate conflicts, hence converging to a normative system, whenever the complaint rate goes beyond 38%. Specifically, given a 38% complain rate, IRON converges between 78% and 86% of the times as the norm infringement rate increases. Finally, beyond 42% complaint rate, IRON converges 100% of the times. Overall, IRON synthesised 20 norms, and ended up simulations with a 5-norm normative system. Likewise the case of population 1 (majority of complaining users), IRON converged to normative systems with 75% compactness.

**3. Minority of complaining users.** To conclude, we now analyse IRON's convergence rate for a population of 3 moderates and 7 spammers. Figure 10c illustrates the results. Even though this population has the lowest degree of moderate users, and therefore the lowest capability to detect conflicting contents, IRON starts generating norms when moderate users complain about 58% of the contents. Specifically, for a 58% complaint rate, IRON converges between 8% and 11% of the simulations, and for 62% complaint rate it converges between 82% and 88% of the total simulations. Finally, for simulations with a complaint rate equal or higher than 65%, IRON converges 100% of the simulations for each norm infringement rate. Overall, IRON synthesised 28 norms, and ended up simulations with a 7-norm normative system. Likewise the cases of populations 1 and 2, IRON converged to normative systems with 75% compactness.

As a summary, these experiments demonstrate that IRON manages to synthesise very compact normative systems by means of generalisations. Specifically, for the three populations that we evaluated, IRON converged to normative systems with 75% compactness.

[3] We have chosen to show the results for 38% complaint rate to capture the transition area where IRON passes from 0% convergence to 100%.

## 6. Related Work

Regulating users' behaviours in on-line communities is a complex problem that has been tackled through several approaches. Most on-line communities aim at regulating communities by: (i) imposing some predesigned terms and conditions to the users; and (ii) employing human moderators who perform corrective actions. As an example, *FansCup* [8] is a soccer on-line community that bases its regulation on human moderators. In particular, moderators are users of the community promoted by the community managers. To facilitate moderation tasks, *FansCup* provides moderators with a moderator's manual that describes what actions to perform in some situations. Other large on-line communities externalise some moderation tasks, assigning them to companies that offer moderation services. As an example, *eModeration* [9] and *Crisp* [6] are companies that offer moderation services to on-line communities. For instance, *Yahoo* employs *Crisp* to externalise some moderation tasks. In general, the terms and conditions used by moderators are set by the owner of the on-line community without users' participation.

Most current scientific research in on-line communities has focused on studying users' dynamics and behaviours within on-line communities. Some works use computational frameworks to model and simulate users behaviours and interactions in on-line communities. The work in [19] uses an agent-based approach, the so called VLCs (Virtual Learning Communities simulator), to study the relationship between the individual behaviour of participants and the overall development of an on-line community. The authors perform an empirical evaluation where they discuss general observations on users' behaviours, based on a series of comparative simulations. In [11] the authors present *Comtella*, a framework to model on-line communities with a system dynamics approach to simulate the overall behaviours of participants in on-line communities. However, this simulator does not take into account the feedback effects on content quality. The work in [16] presents COSIMO, a simulator to predict the behaviour in an on-line community for different policies which are pre-established by a designer. The authors employ real data from *Lycos* [10] to empirically evaluate COSIMO, studying how different policies affect the quality of the contents in the community. However, to the best of our knowledge, no previous work has tackled the automatic synthesis of norms for on-line communities.

Regarding the synthesis of norms for coordinating Multi-Agent Systems (MAS), we differentiate two strands of work tackling this problem: *off-line* and *on-line* approaches. On the one hand, off-line approaches (such as [17,3]) aim at synthesising norms for a MAS that constrain the behaviour of agents while ensuring the achievement of global system goals. Off-line approaches require detailed knowledge of a MAS (i.e., its full state space) at design time. Following [17], the complexity of the norm synthesis problem is high (NP-complete). This has recently spurred research to better cope with the size of the state space [2].

Nonetheless, off-line design is not appropriate to cope with open, dynamic MAS, whose agents' population, composition and state space change with time. On-line norm synthesis approaches try to overcome such limitations by synthesising norms that regulate a MAS at run-time instead of at design time. More recently, norm emergence has become a popular technique for on-line norm synthesis. Norm emergence approaches (such as [15]) do not require any global state representation or centralized control. Instead, it considers that agents collaboratively choose their own norms out of a space of possible norms. A norm is considered to have emerged when a majority of agents adopt it and abide by it. The work in [5] explores characteristics that affect the longevity and adoption of emerged norms in a tag-based cooperation scenario. Other works like [18] focus on social instruments like rewiring and observation to facilitate the emergence of norms from repeated interactions between the individuals in a society. In [14], the authors design a spreading-based convention emergence mechanism that helps agents to agree, in a distributed manner, on the best convention when there are multiple alternatives. Nevertheless, approaches based on norm emergence are highly sensitive to the initial conditions in the MAS. Moreover, norm emergence assumes that agents are endowed with the necessary machinery to participate in the emergence process. Other works like [7] focus on norm compliance checking. Briefly, in this work the authors model a set of interrelated norms as Norm Nets, and then map them to Coloured Petri Nets to check whether agents can comply with the norms imposed on them.

Against this background, our work in [12] proposes a novel mechanism called IRON (Intelligent Robust On-line Norm synthesis machine) for the on-line synthesis of norms. IRON produces norms for the agent population in a MAS that characterise necessary conditions for coordination, while avoiding over-regulation. However, [12] does not take into account agents' feedback to identify conflicts, unlike we do in this paper.

## 7. Conclusions and Future Work

In this paper we have applied IRON, a novel mechanism for the automated synthesis of normative systems, to a particular on-line community scenario, where agents model users that share contents within an on-line community. First, we have presented an on-line communities simulator that provides the MAS scenario for IRON to regulate. Our simulator allows to design different populations of users with different user profiles, which establish the frequency and type of contents that users upload, view and complain about. Moreover, it provides visualisation tools that allow to analyse the norm synthesis process. In particular, the scenario display allows to visualise agents' interactions in the community. Second, we have described the process to configure and exploit IRON to synthesise norms for our on-line community. With this aim, we have described our domain-specific implementation of IRON inputs. As a result, IRON synthesises norms for the users of the on-line community, preventing them from uploading inappropriate contents, and hence avoiding frictions between users. Third, we have performed an empirical evaluation to study IRON's convergence. We have first performed a micro analysis of IRON's convergence process, showing how it manages to synthesise norms based on users' complaints that avoid new conflicts. We have also performed a macro analysis to analyse IRON's convergence rate for different populations, as well as complaint rates and norm infringement rates. Initial results demonstrate that for medium and high complain and norm infringement rates, IRON is capable of synthesising norms that avoid frictions in the on-line community scenario.

As future work, we plan to extend the simulator to include punishments for those agents that do not comply with norms. We also aim at performing large-scale experiments that capture more realistic situations. Additionally, we also plan to implement a basic on-line community in order to perform experiments with humans. Thus, our final aim is to apply IRON's norm synthesis to actual-world on-line communities, automatically synthesising norms for humans.

## References


[1] A. Aamodt and E. Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1):39–59, 1994.

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

[3] D. Fitoussi and M. Tennenholtz. Minimal social laws. In *Proceedings of the National Conference on Artificial Intelligence*, pages 26–31. John Wiley & Sons LTD, 1998.

[4] A. García-Camino, J. A. Rodríguez-Aguilar, C. Sierra, and W. Vasconcelos. Constraint rule-based programming of norms for electronic institutions. *Autonomous Agents and Multi-Agent Systems*, pages 186–217, 2009.

[5] N. Griffiths and M. Luck. Norm Emergence in Tag-Based Cooperation. In *9th International Workshop in Coordination, Organizations, Institutions and Norms (COIN), in the International Conference on Autonomous agents and multi-agent systems (AAMAS'10). 79-86*, 2010.

[6] A. Hildreth. Crisp. `http://www.crispthinking.com`, 2005.

[7] J. Jiang, V. Dignum, H. Aldewereld, F. Dignum, and Y.-H. Tan. Norm compliance checking. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 1121–1122. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[8] P. Lanas and D. Garzón. FansCup. `http://www.fanscup.com`, 2005.

[9] Littleton, Tamara. eModeration. `http://www.emoderation.com`, 2002.

[10] M. Loren Mauldin. Lycos. `http://www.lycos.com`, 1995.

[11] Y. Mao, J. Vassileva, and W. Grassmann. A system dynamics approach to study virtual communities. In *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on*, pages 178a–178a, 2007.

[12] J. Morales, M. Lopez-Sanchez, J. A. Rodriguez-Aguilar, M. Wooldridge, and W. Vasconcelos. Automated synthesis of normative systems. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, AAMAS '13, pages 483–490, 2013.

[13] M. North, N. Collier, J. Ozik, E. Tatara, C. Macal, M. Bragen, and P. Sydelko. Complex adaptive systems modeling with repast simphony. *Complex Adaptive Systems Modeling*, 1(1):3, 2013.

[14] N. Salazar, J. A. Rodriguez-Aguilar, and J. L. Arcos. Robust coordination in large convention spaces. *AI Communications*, 23(4):357–372, Dec. 2010.

[15] B. Savarimuthu, S. Cranefield, M. Purvis, and M. Purvis. Role model based mechanism for norm emergence in artificial agent societies. *Lecture Notes in Computer Science*, 4870:203–217, 2008.

[16] F. Schwagereit, S. Sizov, and S. Staab. Finding optimal policies for online communities with cosimo. *Proceedings of the WebSci10: Extending the Frontiers of Society On-Line, April*, 2010.

[17] Y. Shoham and M. Tennenholtz. On social laws for artificial agent societies: off-line design. *Journal of Artificial Intelligence*, 73(1-2):231–252, February 1995.

[18] D. Villatoro, J. Sabater-Mir, and S. Sen. Social instruments for robust convention emergence. In T. Walsh, editor, *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 420–425. IJCAI/AAAI, 2011.

[19] Y. Zhang and M. Tanniru. An agent-based approach to study virtual learning communities. In *System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on*, pages 11c–11c. IEEE, 2005.