Applying IRON to a Virtual Community Scenario [1]

Javier Morales [a,b] and Iosu Mendizabal [c] and David Sanchez-Pinsach [b] and Maite Lopez-Sanchez [b] and Juan A. Rodriguez-Aguilar [a]

[a] *Artificial Intelligence Research Institute (IIIA-CSIC). Bellaterra, Spain.*
[b] *MAiA Department, Universitat de Barcelona. Barcelona, Spain.*
[c] *University of Mondragon. Mondragon, Spain*

Abstract. Normative systems (norms) have been widely proposed as a technique for coordinating multi-agent systems (MAS). The automated synthesis of norms is a complex problem that remains open. IRON (Intelligent Robust On-line Norm synthesis mechanism) is a novel mechanism for the on-line automated synthesis of norms for MASs. IRON produces conflict-free norms that characterise necessary conditions for coordination, without over-regulation. In the past, IRON successfully regulated a traffic scenario even in the presence of non-compliant agents. In this paper, we apply IRON to synthesise norms for a virtual community scenario, where agents are users that share contents within the community. As a result, IRON synthesises norms that prevent users from uploading undesirable contents (i.e., those that users complain about).

Keywords. multi-agent normative systems, virtual communities

1. Introduction

Norms have been widely proposed as a technique for coordinating multi-agent systems (MAS). A norm can be understood as an established, expected pattern of behaviour [18]. Typically, these behavioural patterns impose constraints on the behaviour of individuals in order to avoid conflicts (e.g., user complaints in a virtual communities scenario).

Since the seminal work of Shoham and Tennenholtz [16], the problem of norm synthesis (i.e., determining the set of norms that avoid conflicting states) has attracted considerable attention. We differentiate two strands of work tackling this problem: the *off-line* and *on-line* norm synthesis approaches. On the one hand, off-line approaches (such as [16,5]) 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. Some refinements to the basic approach have included the implementation costs of norms and multiple design goals with different priorities

[1] This work was funded by AT (CONSOLIDER CSD2007-0022), EVE (TIN2009-14702-C02-01/02), COR (TIN2012-38876-C02-01/02), MECER (201250E053) and the Generalitat of Catalunya (2009-SGR-1434).

[2]. Following [16], 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 [3].

Nonetheless, off-line design is not appropriate to cope with open MAS, whose composition and state space change with time. On-line norm synthesis approaches (such as [13]) 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 (e.g., [8,12,14,15,17]). It does 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. Nevertheless, approaches based on norm emergence suffer from several drawbacks. Firstly, convergence is highly sensitive to the initial conditions in the MAS. Secondly, there is the assumption that agents collaborate during the norm synthesis process and that agents are endowed with the necessary machinery to participate in the emergence process. Third, regarding the norm synthesis process, although the utility of norms is eventually considered, there are further aspects that, to the best of our knowledge, have not been taken into account yet. On the one hand, it is not considered whether a synthesised norm is truly necessary or not (because, for example, its regulation is subsumed by another norm). Thus, it might be the case that a norm within a normative system (the set of active norms in the MAS) is not really necessary, and hence it leads to over-regulation: agents must handle more norms than needed. On the other hand, the generalisation of a set of norms into a more general one is not considered either as part of the norm synthesis process.

Against this background, the work in [10] 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. IRON synthesises norms that are both *effective* and *necessary*. On the one hand, *effectiveness* computes how norms manage to avoid conflicts whenever agents comply with these norms. On the other hand, *necessity* computes to what extent norms are necessary to regulate conflicts whenever agents do not comply with them. Furthermore, IRON is endowed with the capability of generalising norms. By generalising norms and discarding unnecessary norms, IRON allows IRON to yield *concise* normative systems.

In previous work [10], IRON was empirically evaluated in a traffic junction scenario where agents were travelling cars and the goal of the MAS was to avoid collisions between cars. These empirical results showed that IRON automatically synthesised concise normative systems that avoided collisions despite a high percentage of violations (up to 50%). In this work, we use IRON to synthesise norms for a simulated virtual community scenario, where agents are users that share contents within a virtual community. The goal of the MAS is to avoid conflicting situations that make users to feel uncomfortable within the community. These conflicting situations can be identified based on the complaints users report about conflicting contents. Consequently, resulting synthesised norms prevent users from uploading conflicting contents to the virtual community. The contributions of this

paper are *(i)* a simulator to simulate the interactions of users within a virtual community, and *(ii)* the application of IRON to a virtual community scenario.

Regarding virtual communities, the work in [9] presents *Comtella*, a framework to simulate the overall behaviours of participants in the communities. Other works like [4] study how to refine norms in a virtual community by observing individuals' behaviour regarding a set of pre-defined norms. However, to the best of our knowledge, no previous work has tackled the automatic synthesis of norms for virtual communities.

The paper is thus organised as follows. Next Section 2 briefly introduces IRON. Section 3 describes our virtual community simulator so that Section 4 can detail the application of IRON to the synthesis of norms for this virtual community scenario. Finally, Section 5 draws some conclusions and sets paths to future research.

2. Background

In this section we survey the Intelligent Robust On-line Norm synthesis mechanism (IRON), a norm synthesis approach aimed at synthesising effective (conflict-avoiding) normative systems for MASs. IRON is based on four main components: (i) a grammar to synthesise new norms; (ii) the normative network (a data structure to represent normative systems and explored norms); (iii) a set of operators that make it possible to transform one normative system into another; and (iv) a strategy that specifies when to use such operators. We describe below each component together with IRON's architecture.

2.1. A Grammar for norm synthesis

IRON employs a grammar to synthesise norms. Norms are constructs of the form $\langle \varphi, \theta(Ac) \rangle$, where φ is the precondition of the norm and $\theta(Ac)$ is a deontic operator that establishes an obligation or prohibition over an action in the set Ac of agents' available actions. As mentioned above, norms are described from a local individual's perspective. Hence, whenever the local perception of an agent satisfies the precondition φ of a norm, then the obligation or prohibition described by $\theta(Ac)$ holds for it. IRON adapts its grammar from [6], 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 $\mathcal{L}_{Ag}$.

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

2.2. The Normative Network

IRON uses the so called Normative Network: a data structure that represents normative systems and explored norms. Specifically, a Normative Network is a graph-based data structure where nodes stand for norms and whose relationships stand for (generalisation) relationships among them. Norms can be *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.

2.3. Operators for normative networks

IRON includes four different operators $O = \{create, deactivate, generalise,$ and $specialise\}$ to transform the normative network, leading from one normative system into another. Operator *create* synthesises a new norm and adds it to the normative network. Operator *deactivate* sets the state of the norm as *inactive* in the normative network, hence removing it from the normative system. Operator *generalise* joins several norms into a more general unique norm (hence reducing the cardinality of the normative system). Finally, operator *specialise* undoes a norm generalisation, splitting a general norm into a set of more specific norms.

2.4. IRON*'s strategy*

Previous operators are invoked by following a specific strategy. The strategy that IRON follows to perform the norm synthesis is as follows. Given a MAS, IRON operates by continuously iterating the following steps: (1) it monitors the MAS operation through its *sensors*, searching for conflicts. It represents perceived agents' interactions in the form of *views*, which are descriptions of the scenario from a global, external observer's perspective; (2) whenever a new non-regulated conflict is detected, a *norm generation* process is carried out, which generates a new norm aimed at avoiding that conflict in the future. This generation process is based on an unsupervised version of classical Case-Based Reasoning (CBR) [1]. IRON synthesises norms to regulate agents' behaviour. Therefore, since agents must be able to understand norms, IRON describes norms from an agent local perspective; (3) next, it performs a *norm evaluation* process, which computes the performance (in terms of effectiveness and necessity) of the norms that have been applied and violated during current time step; (4) afterwards, it applies a *norm refinement* process that generalises, specialises or even discards norms according to their effectiveness and necessity ranges during a given period of time T (decisions are made based on specific thresholds); (5) finally, if the normative system has been changed, IRON sends it to the agents in the regulated MAS scenario.

Specifically, the proposal of [10] is to monitor the evolution of the system at regular time intervals (i.e., ticks) and apply operators under certain conditions. Initially, during the synthesis of new norms, for each detected conflict, the strategy invokes operator *create* which generates a new norm aimed at avoiding the conflict in the future. The new norm is added it to the normative network and its state is set to *active*. Afterwards, the norm evaluation process retrieves those norms that have been applied and violated during the current step. Moreover, it also determines which norms led to conflicts during the current time step. As a result, it obtains a partition of applicable norms into: *(i)* applied norms that led to conflicts; *(ii)* applied norms that did not lead to conflicts; *(iii)* violated norms that led to conflicts; and *(iv)* violated norms that did not lead to conflicts. Then, the strategy computes: the effectiveness of norms at current time step t ($\mu_{eff}(n,t)$) by considering successful applications (i.e., applications not leading to conflicts) at time step t; and their necessity at current time step t ($\mu_{nec}(n,t)$) based on harmful violations (violations leading to conflicts). These effectiveness and necessity values ($\mu_{eff}(n,t)$ and $\mu_{nec}(n,t)$) are then aggregated over a period of time T to compute

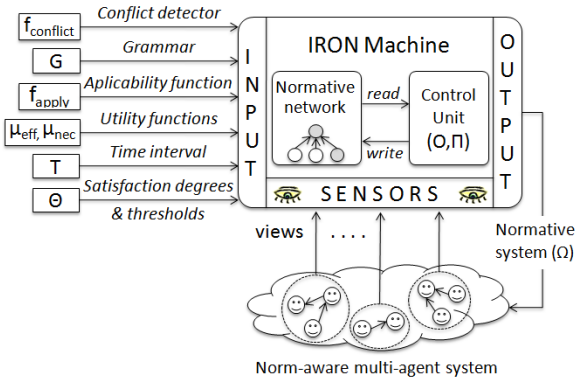

Figure 1. IRON's architecture.

$\mathcal{E}(n,T)$ and $\mathcal{N}(n,T)$, the corresponding performance ranges. Finally, the norm refinement process yields a new normative system by transforming the normative network via deactivating ineffective or unnecessary norms, and performing norm generalisations and specialisations. Specifically, if the effectiveness or the necessity range of a norm has been under a specialisation threshold α_{spec} during a period of time T, then the norm is specialised (or deactivated if the norm does not generalise any other norm) by invoking the *specialise* (or *deactivate*) operator. If that is not the case, then the strategy tries to generalise the norm (by means of the *generalise* operator) whenever its effectiveness and necessity ranges have been over a generalisation threshold α_{gen} during the time period T.

2.5. IRON's architecture

Figure 1 depicts the architecture that [10] proposes for performing this norm synthesis process. IRON is an abstract, domain independent mechanism for the synthesis of norms. However, in order to apply IRON to different scenarios, it requires some domain-dependant inputs: (i) a function $f_{conflict}$ to detect conflicts in agents interactions; (ii) a grammar $\mathcal{G}$ to define norms; (iii) a function f_{apply} to determine whether a norm applies to the agents in a given view; (iv) evaluation functions to compute the effectiveness (μ_{eff}) and necessity (μ_{nec}) of norms in the normative network; (v) Θ, a set of satisfaction degrees and thresholds that define the acceptable range of effectiveness and necessity for norms; as well as (vi) the time interval (T) to consider.

3. The Virtual Communities Simulator

This section introduces our Virtual Communities Simulator. It is based on *Repast Simphony* [11] and allows to perform discrete agent-based simulations of virtual communities. Within a virtual community, agents represent users that interact and share different types of contents.

Our simulator represents a virtual community as a grid (see bottom of Figure 2) where each cell corresponds to a position where a user can upload a content. Columns in the grid are grouped into three different content sections: *Forum*, *Mul-*

timedia and *The Reporter*. Each row corresponds to an agent in the virtual community user population. The simulator has a pre-defined set of contents for each user. Thus, whenever a user uploads a new content to a section of the community, it is displayed into the next empty cell for that section in the corresponding user's row. Additionally, contents are assumed to be previously categorised, so that we distinguish contents to be correct (displayed in green in the figure) from (red coloured) conflicting ones, which can actually correspond to spam, troll, rude, or violent contents. Users are considered to be moderate or conflictive depending on the majoritarian content type they upload. Moderate users (displayed with an M label) are modelled to complain about conflicting contents. Complaints for conflicting contents are displayed below them in blue as exclamation sign labels.

Briefly, the simulator works as follows. At each tick, community users: (1) upload new contents to some section of the virtual community; (2) view some uploaded contents; and (3) complain about seen contents that are considered to be conflictive (this action is just performed by moderate users). Each user has a *user profile* that describes how often they upload, view and complain about contents. Moreover, it also describes what sections and type of contents they choose to upload, view and complain about. According to the type of contents that they upload, users can be divided into six main types: *moderate, spammer, troll, pornographic, violent* and *rude*. While moderate users upload correct contents, spammer, trolls, pornographic and violent users tend to upload contents that lead to user complaints, decreasing the overall user satisfaction of the community.

Our simulator provides some components to configure, execute and monitor simulations: (i) an agents' population design tool that allows to design populations of agents with different user profiles; (ii) a scenario display (shown in Figure 2) that allows to visualise the virtual community, as well as the contents that users upload and their associated complaints; and (iii) a set of tools to analyse the results of simulations, including statistical tools based on *JFreeChart* [7] that generate charts as well as output file functionalities that record the evolution along time of any property of users or contents. Next, we describe the tool provided for defining user populations.

3.1. Designing Agents' Populations

Our virtual communities simulator provides a tool that allows to design different user (agent) populations. Top of Figure 2 depicts its graphical user interface. A population is composed of different types of agents, which in turn have three interaction profiles that state the frequency of performing actions in the virtual community: *upload content, view content* and *complain about a content*.

The upload profile describes: (1) the upload frequency, which is the probability of the user to upload a content at a given time step (tick); and (2) the probability of the user to upload each one of the six different content types at a given tick. For instance, we may design a pure moderate agent type by assigning 1 to the probability of uploading correct contents and 0 to all the probabilities of uploading conflicting contents (e.g. spam).

The view profile defines users' preferences in terms of the probability to view contents from each section of the community. Moreover, the view profile also considers the *view mode*, which describes three different ways to choose contents to

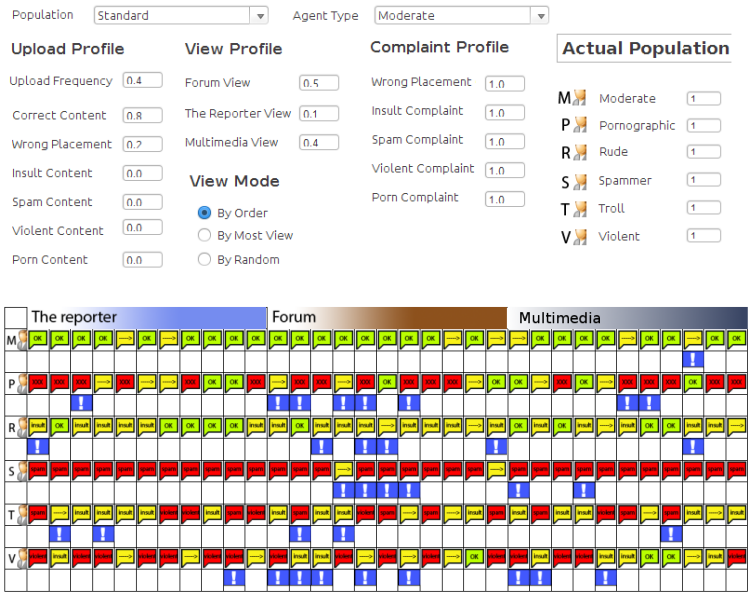

Figure 2. Virtual Communities Simulator: (top) Population design tool; (bottom) Grid representation of the virtual community scenario.

view: *by order* (which chooses last uploaded contents), *most viewed* (which chooses most visited contents and *randomly*, according to a uniform random distribution. **The complain profile** describes the probability of a user to complain about each type of visited content. Thus, to design moderate users we should configure it to complain about any conflicting content with a probability of 1.

4. Synthesising Norms for Virtual Communities

IRON is an abstract, domain independent mechanism for the synthesis of norms. Hence, in order to apply its norm synthesis to a specific scenario, IRON requires the set of domain-dependant inputs enumerated in Section 2. Next subsections provide the specification of these inputs for the virtual community scenario.

4.1. A function for conflict detection

Function $f_{conflict}$ identifies the conflicts in current MAS state. In our virtual community scenario, conflictive situations can be identified based on the complaints users report about contents. Therefore, our implementation of $f_{conflict}$ is devoted to update the *complaint ratio* of all contents that have been viewed at current state ($view$). The complaint ratio of a content is computed as the ratio of accumulated number of complaints over its total number of views. In this manner, the function returns the list of those contents that are considered to be conflictive, or in other words, that have a *complaint ratio* above a conflict threshold $\alpha_{conflict}$:

$$\frac{totalNumComplaints(content)}{totalNumViews(content)} > \alpha_{conflict}$$

4.2. A grammar for norm synthesis

IRON norm synthesis approach assumes that a conflict can be avoided if some of the agent actions that caused it are not performed. Thus, it generates norms that prohibit agents in the same conflictive context to perform such actions.

By following the grammar specification in previous Section 2.1, we instantiate this grammar as follows. On the one hand, a norm precondition describes the local perception (i.e., context) of an agent in the conflict. This context is defined in terms of predicates $p^n(\tau_1, \ldots, \tau_n)$ that in our case happen to be unary $p = \{user, section, contentType\}$. Corresponding terms τ for these predicates are, respectively: all user identifiers (τ=$\{$`user1`, ..., `user10`$\}$); section names (τ=$\{$`Forum`, `Multimedia`, `The Reporter`$\}$); and content types (τ=$\{$`correct`, `spam`, `troll`, `rude`, `violent`$\}$). On the other hand, a norm consequence specifies the prohibition (in this case, deontic operator $\theta = phr$) to perform the action `upload` of the content in the context.

Therefore, norms establish prohibitions for certain users to upload certain types of contents in some section of the community. Thus, whenever a user agent (with a certain id) is visiting a section of the community and it is about to upload a content of a conflicting type, then a norm should apply to the agent which prohibits it to upload that content. Norms n_1 and n_2 are examples of norms that IRON automatically synthesises for our scenario:

$n_1 : \langle (user(\texttt{user1}), section(\texttt{Multimedia}), contentType(\texttt{spam})), prh(\texttt{upload}(content)) \rangle$
$n_2 : \langle (user(\texttt{user3}), section(\texttt{Forum}), contentType(\texttt{violent})), prh(\texttt{upload}(content)) \rangle$

Norm n_1 prohibits user `user1` to upload spam contents on the Multimedia section, and norm n_2 prohibits user `user3` to upload violent contents on the Forum section.

4.3. A function for detecting norm applicability

During the norm evaluation phase, IRON retrieves the norms that have been applied and violated during the current time step. With this aim, IRON requires as an input function f_{apply} to detect the norms that apply to the agents at a given state of the MAS.

Our implementation of the applicability function works as follows: given a state of the MAS, it retrieves the agents that interact within it. Next, it interprets the local perceptions of each agent in the state. Finally, it retrieves the norms that apply to the local perception of each agent. As an example, consider norm n_1 described above. Consider now that at a given state of the MAS at time t, a user with id `user1` uploads a `spam` content to section `Multimedia`. Formally, its local perception is:

$(user(\texttt{user1}), section(\texttt{Multimedia}), contentType(\texttt{spam}))$

Given this particular MAS state, function f_{apply} will first retrieve `user1`, which is the agent that interacts with the community in it. Next, it will interpret its local perception at the given state. Finally, the applicability function will retrieve the

norms that apply to the user. In particular, the local perception of user `user1` satisfies the precondition of norm n_1. Therefore, at the given MAS state at time t, norm n_1 applies to user `user1`.

4.4. Time interval and thresholds

As explained in section 2, during the norm evaluation phase IRON refines the normative system deactivating, generalising and specialising norms, based on their effectiveness and necessity ranges over a period of time T, and a set of thresholds. In our particular scenario we have established a time interval $T = 100$, configuring IRON's to compute effectiveness and necessity ranges with a great number of punctual values along time. Regarding the thresholds, we have taken a conservative approach to just generalise norms that really perform well (have a high effectiveness and necessity), and to deactivate norms that really perform poorly. Thus, the generalisation threshold has been set with a high value ($\alpha_{gen} = 0.6$), and the specialisation threshold has been set to a low value ($\alpha_{spec} = 0.2$).

4.5. An execution of IRON*'s norm synthesis*

As a proof of concept of the application of IRON to the virtual community scenario, in this section we comment on a typical execution of the norm synthesis that IRON performs for it. In particular, we show how, given a population of community users where the great majority are moderate and some of them are spammers, IRON synthesises norms that prohibit spammers to upload conflicting contents.

Consider a population of 10 users where 8 users (with id's $\{\texttt{user1}, \ldots, \texttt{user8}\}$) are moderate and 2 users (with id's `user9` and `user10`) are spammers. On the one hand, moderate users upload correct contents with a probability of 0.8, and wrong placed content with a probability of 0.2. Moreover, moderate users complain about spam with a probability of 0.7. On the other hand, spammers upload spam with a probability of 0.7 and correct contents with a probability of 0.3. While executing a simulation with this population, IRON rapidly manages to automatically synthesise a normative system that contains norms prohibiting users 9 and 10 (spammers) to upload spam contents to any section of the virtual community, hence avoiding complaints of moderate users:

$n_1 : \langle (user(\texttt{user9}), contentType(\texttt{spam})), prh(\texttt{upload}(content)) \rangle$
$n_2 : \langle (user(\texttt{user10}), contentType(\texttt{spam})), prh(\texttt{upload}(content)) \rangle$

5. Conclusions

In this paper we have applied IRON, a novel mechanism for the automated synthesis of normative systems, to a particular virtual community scenario, where agents are users that share contents within the on-line community. Firstly, we have presented a virtual 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. Secondly, we have applied IRON to this particular scenario. For this purpose, we have described our domain-specific

implementation of IRON inputs. As a result, IRON synthesises norms for the users of the virtual community, preventing them from uploading the type of contents that other users complain about.

As future work, we plan to further evaluate IRON's norm synthesis for the virtual community scenario, studying the degree of convergence and the quality of the resulting normative systems. Additionally, we also plan to extend the simulator to include punishments for those agents that do not comply with norms.

References

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

[2] T. Agotnes and M. Wooldridge. Optimal Social Laws. In *Proceedings of the AAMAS2010*, pages 667–674, 2010.

[3] 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.

[4] C. Danis and A. Lee. The negotiation of norms in an online community, 2002.

[5] 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.

[6] A. García-Camino, J. A. Rodríguez-Aguilar, C. Sierra, and W. Vasconcelos. Constraint rule-based programming of norms for electronic institutions. *JAAMAS*, 2009.

[7] D. Gilbert and T. Morgner. JFreeChart, 2003 - 2005.

[8] N. Griffiths and M. Luck. Norm Emergence in Tag-Based Cooperation. In *9th International Workshop COIN, at AAMAS2010. 79-86*, 2010.

[9] 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.

[10] 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. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[11] 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.

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

[13] 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.

[14] O. Sen and S. Sen. 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, pages 211–222, 2010.

[15] S. Sen and S. Airiau. Emergence of norms through social learning. In *Proceedings of the 20th international joint conference on Artifical intelligence*, IJCAI'07, pages 1507–1512, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[16] 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.

[17] 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.

[18] M. Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley & Sons, 1st edition, June 2002.