Minimality and Simplicity in the On-line Automated Synthesis of Normative Systems[*]

Javier Morales[1,2] and Maite Lopez-Sanchez[1]
Universitat de Barcelona,Spain
jmorales@iiia.csic.es
maite@maia.ub.es

Juan A. Rodriguez-Aguilar
[2]Artificial Intelligence Research Institute (IIIA)
Spanish Council of Scientific Research (CSIC)
Campus UAB. Bellaterra, Spain
jar@iiia.csic.es

Michael Wooldridge
Dept. of Computer Science
University of Oxford
Oxford, UK
mjw@cs.ox.ac.uk

Wamberto Vasconcelos
Dept. of Computing Science
University of Aberdeen
Aberdeen, UK
wvasconcelos@acm.org

ABSTRACT

Much previous research has investigated explicit, machine-processable norms as a means to facilitate coordination in open multi-agent systems. This research can typically be classified as considering either offline design (norms are synthesised at design time) or online design. Online synthesis techniques aim to construct norms for a system while that system is actually running. A promising recent approach to on-line norm synthesis has been proposed but it suffers from serious drawbacks: (i) it needs too much information; (ii) it ignores issues of compactness in terms of minimality (ensuring that norms are not superfluous) and simplicity (ensuring that agents can process norms with little computational effort). To overcome these drawbacks, we propose an optimistic approach which, even though it uses less information, is able to explore more norms and synthesises sets of norms which are more compact. We present experimental evidence of the quality of our approach.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence —*Multiagent Systems*

Keywords

Norms; Normative Systems; On-line Norm Synthesis

1. INTRODUCTION

Norms are a widely-used technique for coordinating interactions between the individuals of human societies, and researchers in Multi-Agent Systems (MAS) have investigated the possibility of using norms as a technique for coordinating agent interactions in multi-agent systems [3, 2]. However, the problem of computing a normative system (i.e., a set of norms) that will effectively coordinate a multi-agent systems is a computationally complex (NP-hard) problem [10]. Two approaches for norm synthesis have been considered in the literature: off-line and on-line. Off-line approaches (such as [10]) aim to obtain normative systems at design time, requiring complete knowledge of the system state space and all possible agent behaviours. Off-line approaches are thus not appropriate for *open* systems, for which the state space and agent behaviours may not be known. In contrast, on-line approaches aim to synthesise the normative system at runtime. Recently, *norm emergence* has become a popular technique for the run-time synthesis of norms (e.g., [9, 8, 5, 11]). Norm emergence considers that agents collaborate to synthesise their own norms. This assumes that agents will participate in the norm synthesis process, and requires that agents have the appropriate computational capabilities to synthesise norms. An alternative on-line approach is described in [6]. In this work, norms are synthesised at runtime from the observation of agents' interactions, hence not requiring their active participation in the synthesis process. Norm synthesis in [6] is *conflict-driven*: norms are synthesised based on conflicts that arise in the system, where conflicts are considered to be undesirable states of the system. Norms are then continuously evaluated based on the conflicts that arise once agents comply (or not) with them.

However, there are further criteria that are important in the synthesis process. Fitoussi and Tennenholtz [4] introduced *minimality* and *simplicity* as criteria for the off-line synthesis of norms. Minimality is concerned with minimising the number of constraints in a normative system imposed on agents. The intuition is that norms should ideally be as "light touch" as possible, and thus minimality can be seen as attempting to avoid over-regulation. Simplicity is concerned with synthesising norms that are *easy* to reason about by agents. Although it is not explicitly mentioned, in [6] minimality is pursued by discarding norms that are found to be unnecessary, hence reducing the number of constraints imposed on agents.

Building on [6], Morales *et al* developed IRON (*I*ntelligent *R*obust *O*n-line *N*orm synthesis mechanism), an on-line domain-independent architecture and computational model for norm synthesis in MASs [7]. IRON performs norm generalisations, hence synthesising general (abstract) norms that concisely represent groups of (more specific) norms. For instance, in a road traffic scenario, a norm such as "give way to emergency vehicles" is more general, minimal, and simple than having a separate norm for every different type of emergency vehicle (e.g., police, ambulance, fire brigade). In this way, IRON implicitly pursues minimality and simplicity by means of norm generalisations.

However, IRON has some limitations. First, it is highly conservative, since it requires full evidence to generalise norms. As an example, IRON will never synthesise a norm like "give way to emergency vehicles" until it has first synthesised a norm for every kind of emergency vehicle. Consider IRON gathers evidence that not

[*]Thanks to Eva Armengol for her contributions, and to projects AT (CSD2007-0022), COR (TIN2012-38876-C02-01, TIN2012-38876-C02-02), and 2009-SGR-1434. M. Wooldridge was supported by the ERC under Advanced Grant 291528 ("RACE").

Appears in: *Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (eds.), Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), May 5-9, 2014, Paris, France.*

giving way to police and ambulances may lead to conflicting situations (e.g., collisions), but it never gathers evidences to consider that not giving way to fire-brigades may be prejudicial. Therefore, it will synthesise norms to give way to police and ambulances in order to avoid conflicts, but will never synthesise norms to give way to fire-brigades. As a consequence, it will never be able to synthesise a compact, general norm to give way to *all* emergency vehicles (police, ambulances, *and* the fire-brigade), inhibiting the minimality and simplicity of the normative system. Another shortcoming is that minimality and simplicity are not explicitly addressed in IRON's mechanisms or experiments.

An alternative method for generalisation (in the context of logical reasoning) is described in [1]. It describes the *anti-unification* method, which consists of generalising feature terms to their *least common subsumer* or *most specific generalisation*, generalising a set of terms to the most specific term that is common to all of them.

The main contribution in this paper is SIMON (*SI*mple *M*inimal *O*n-line *N*orm Synthesis), a new approach to norm synthesis, its generalisation inspired by [1], which significantly outperforms the approach of [7] with respect to the key criteria of minimality and simplicity. SIMON adopts an *optimistic* approach for norm generalisation, managing to synthesise more compact normative systems.

The remainder of this paper is organised as follows. Section 2 gives the background and concepts on synthesis of normative systems required for our approach, which is presented in Section 3. Section 4 reports our empirical evaluation, and Section 5 concludes.

2. BACKGROUND & CONCEPTS

We now present the key concepts of IRON [7], extending and adapting some of its definitions and using new examples to facilitate comparison with our proposed approach. IRON is an on-line mechanism aimed at synthesising norms to avoid conflicts in multi-agent systems. In brief, it works by continuously monitoring a system, searching for undesirable states (conflicts). Whenever such conflicts arise, IRON synthesises new norms to resolve these conflicts. These new norms are then communicated to the agents within the system with the aim of preventing these conflicts from occurring again in the future. Agents can choose whether to comply or not with norms and IRON monitors the effect of such decision (checking, for example, whether conflicts still arise or not) to evaluate norms. In other words, after computing new norms, it carries out a *norm evaluation* process that computes the performance of those norms that have been applicable to agents during the current time period. Finally, it carries out a *norm refinement* process, which: (i) generalises norms when possible, joining several norms to a unique parent that concisely represents all of them; and (ii) discards those norms that have not performed well for a period of time.

To facilitate comparison with our approach, we adopt IRON's information model and consider a normative MAS composed of a set of agents Ag, and a finite set of actions $Ac = \{ac_1, \ldots, ac_m\}$ available to them. Agents describe their *local perception* of the world in terms of a logical language $\mathcal{L}_{Ag}$. Norms are of the form $\langle \varphi, \theta(ac) \rangle$, where φ is the precondition of the norm, θ is a deontic operator (e.g., a *prohibition*) and ac is an action available to agents. The precondition φ of a norm is a set of first-order predicates $p(\tau_1, \ldots, \tau_n)$, where p is a predicate symbol and $\tau_1, \ldots, \tau_n$ are terms of the language $\mathcal{L}_{Ag}$. Whenever the *local perception* of an agent satisfies the precondition of a norm, then the norm applies to the agent and the deontic condition $\theta(ac)$ holds for this agent. A *normative system* (NS) corresponds to the current set of active norms for an agent.

We now introduce a running example to be used throughout the remainder of the paper. We consider a traffic scenario, with unary

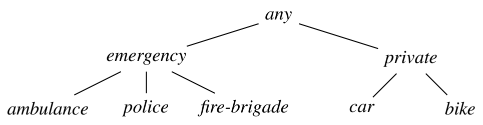

Figure 1: Relationships between terms

predicate symbols {*left*, *front*, *right*} representing the three road positions that an agent perceives. Each predicate has a single term from those depicted in Figure 1. We capture taxonomic knowledge relating terms in our domain, this knowledge being essential to our approach. Specifically, terms *ambulance*, *police* and *fire-brigade* represent types of *emergency* vehicles, while the terms *car* and *bike* represent types of *private* vehicles. Finally, the term *any* stands for any kind of vehicle, whether *emergency* or *private*. The actions available to agents are $Ac = \{go, stop\}$. With these definitions in place we can create norms such as $\mathbf{n}_1, \ldots, \mathbf{n}_6$ as described below. For example, $\mathbf{n}_1$ prohibits an agent from *go*ing if it perceives an *ambulance* vehicle to its left, a *police* car in front, and a *car* to its right. The remaining norms address different circumstances.

$$\begin{array}{ll}
\mathbf{n}_1 : \langle\{\mathit{left}(\mathit{ambulance}), & \mathit{front}(\mathit{police}),\ \mathit{right}(\mathit{car})\}, \quad \mathit{prh}(\mathit{go})\rangle \\
\mathbf{n}_2 : \langle\{\mathit{left}(\mathit{police}), & \mathit{front}(\mathit{police}),\ \mathit{right}(\mathit{car})\}, \quad \mathit{prh}(\mathit{go})\rangle \\
\mathbf{n}_3 : \langle\{\mathit{left}(\mathit{fire\text{-}brigade}), & \mathit{front}(\mathit{police}),\ \mathit{right}(\mathit{car})\}, \quad \mathit{prh}(\mathit{go})\rangle \\
\mathbf{n}_4 : \langle\{\mathit{left}(\mathit{fire\text{-}brigade}), & \mathit{front}(\mathit{police}),\ \mathit{right}(\mathit{police})\}, \ \mathit{prh}(\mathit{go})\rangle \\
\mathbf{n}_5 : \langle\{\mathit{left}(\mathit{emergency}), & \mathit{front}(\mathit{police}),\ \mathit{right}(\mathit{car})\}, \quad \mathit{prh}(\mathit{go})\rangle \\
\mathbf{n}_6 : \langle\{\mathit{left}(\mathit{fire\text{-}brigade}), & \mathit{front}(\mathit{police}),\ \mathit{right}(\mathit{any})\}, \quad \mathit{prh}(\mathit{go})\rangle
\end{array}$$

In order to represent norms, IRON employs a graph-based data structure called a *normative network* (NN). A normative network is a graph whose nodes represent norms and whose edges correspond to generalisation relationships between norms. Norms in the NN may be either *active* or *inactive*, and a normative system (NS) corresponds to the active norms in the NN. Figure 2 illustrates the evolution of a normative network (and its corresponding NS) over times t and $t + 1$. At time t (depicted in Fig. 2.1), the normative network contains four active norms (coloured in white) $\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3, \mathbf{n}_4$, hence representing the normative system NS=$\{\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3, \mathbf{n}_4\}$. At time $t + 1$ (see Figure 2.2), the NN contains two active norms $\mathbf{n}_5, \mathbf{n}_6$ (i.e., NS=$\{\mathbf{n}_5, \mathbf{n}_6\}$) that concisely represent inactive norms $\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3, \mathbf{n}_4$ (coloured in grey).

Changes from one normative system to another are carried out by applying changes to the normative network. With this aim, IRON incorporates a set of normative network *operators* that make it possible to: (i) *create* norms based on conflicts, *activating* and *adding* them to the normative network; (ii) *deactivate* norms that do not perform well, changing their state in the normative network to *inactive* (no longer belonging to the normative system); (iii) *generalise* several norms to a parent norm, resulting in a more compact representation of the normative system; and (iv) *specialise* norms that do not perform well, as a method to backtrack norm generalisations.

IRON creates a new norm when it detects a new conflict, namely an undesirable state. When this is the case, it retrieves the local context that an agent involved in the conflict had at previous time step as well as the action that this agent performed. It then creates a new norm having as a precondition the agent context described in terms of ordered predicates, and as a deontic operator a prohibition of the action that was carried out (see [6] for further details).

IRON continuously evaluates the performance of norms with respect to their *effectiveness* and *necessity* in avoiding conflicts. On the one hand, IRON measures the cumulative effectiveness of a norm from the outcomes of its applications: the higher the number of *successful applications* (applications that did not result in conflict), the more *effective* the norm. On the other hand, it measures the necessity of a norm according to the following principle: the higher the number of *harmful violations* (violations leading to

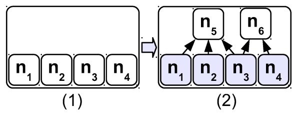

Figure 2: Generalising $\mathbf{n}_1$,$\mathbf{n}_2$,$\mathbf{n}_3$,$\mathbf{n}_4$

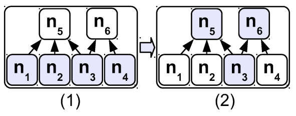

Figure 3: Specialising $\mathbf{n}_5$,$\mathbf{n}_6$

conflicts), the more *necessary* the norm is. Finally, IRON computes the effectiveness and necessity performance ranges of norms during a period of time. These ranges will be essential (as we show below) to perform generalisations and specialisations.

IRON's norm generalisation is based on the computation of potential generalisations and considering *all* the norms in the normative network (even though they are inactive). Firstly, it retrieves all the potential generalisations of each norm that has been evaluated during the current step. They are computed upon the creation of a new norm and consist of all the potential parents this new norm may generalise to, as well as the siblings that are required for each generalisation. For example, in Figure 2.1, when $\mathbf{n}_3$ is created, $\mathbf{n}_5$ and $\mathbf{n}_6$ are computed as its potential parents, where $\mathbf{n}_5$ has $\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3$ as children and $\mathbf{n}_6$ has $\mathbf{n}_3, \mathbf{n}_4$. Secondly, for each potential parent, it checks if all its children: (i) have been synthesised; (ii) are either *active* in the normative network or they are *inactive* but generalised by an active norm; and (iii) they have performed well during a period of time. Specifically, a norm $\mathbf{n}$ has performed well during a period if the lower bounds of its effectiveness *and* necessity ranges are above some satisfaction (generalisation) thresholds. Finally, if all three conditions hold, IRON performs generalisation, synthesising the parent norm (including it in the normative network), deactivating all the children, and establishing generalisation relationships between the children and the parent. Figure 2 illustrates the generalisation of $\mathbf{n}_3$ to its potential parents $\mathbf{n}_5$ and $\mathbf{n}_6$, (by adding $\mathbf{n}_5$, $\mathbf{n}_6$ to the NN and deactivating $\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3, \mathbf{n}_4$). Notice that $\mathbf{n}_5$, $\mathbf{n}_6$ are general, compact representations of $\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3, \mathbf{n}_4$, applicable to agents in specific situations described by $\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3, \mathbf{n}_4$.

As a way to "undo" unwanted norm generalisations, IRON performs norm specialisations whenever norms under-perform. A norm under-performs whenever the higher bound of its effectiveness *or* necessity ranges are below certain thresholds during a period of time. The performance values of general norms are directly affected by the performance values of their children. Therefore, whenever a general norm under-performs, it is due to an under-performing child norm, and thus IRON specialises the general norm by deactivating it and activating all its children but the under-performing one (which remains inactive). Figure 3 depicts the specialisation of norms $\mathbf{n}_5, \mathbf{n}_6$. At time t (Fig. 3.1), the normative network has two active norms $\mathbf{n}_5, \mathbf{n}_6$ that concisely represent norms $\mathbf{n}_1$, $\mathbf{n}_2$, $\mathbf{n}_3$, $\mathbf{n}_4$. Let us suppose $\mathbf{n}_3$ under-performs, so at time $t+1$, IRON deactivates all norms that generalise $\mathbf{n}_3$ (*viz.*, $\mathbf{n}_5, \mathbf{n}_6$) and activates $\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_4$ (Fig. 3.2). Thus, $\mathbf{n}_3$ is no longer represented by the normative system which becomes NS=$\{\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_4\}$.

We argue that IRON suffers from two major limitations. First, IRON misses out on compactness issues: there are no metrics on minimality and simplicity of the synthesised normative system; nor, indeed, does IRON explicitly address these features. Indeed, there is a lack of literature on experimental analysis of compactness, and work so far has only addressed compactness issues from a theoretical perspective [4]. Additionally, IRON's generalisation is highly conservative. IRON requires that *all* the children of a potential generalisation exist, are active and perform well in order to generalise norms. Our approach tackles these drawbacks.

3. ENHANCING NORM SYNTHESIS

We now introduce SIMON (*SI*mple *M*inimal *O*n-line *N*orm Synthesis), our approach to norm synthesis, which overcomes the drawbacks of IRON presented above. SIMON incorporates an alternative technique for norm generalisation which increases the compactness of synthesised normative systems.

3.1 Basic Definitions

We adopt the conventions from [7] to make comparisons easier. Thus, we also consider that the representation language used by agents is denoted by $\mathcal{L}_{Ag}$. Moreover, we will denote the set of terms in the language by $\mathcal{T}$. We define a relationship between the terms in $\mathcal{T}$ such that if $\tau, \tau' \in \mathcal{T}$ and $\tau' \leq \tau$, we say that τ is more general than τ'. The $\leq$ relationship defines a partial order over the elements of $\mathcal{T}$. We refer to the pair $\mathcal{O} = (\mathcal{T}, \leq)$ as the *ontology*, defining a taxonomy over the terms in $\mathcal{T}$. We require that there is a single term $\tau_0 \in \mathcal{T}$, called the *most general term*, which is not generalised by any other term. Furthermore, for any given term $\tau \in \mathcal{T}$, there is exactly one (possibly empty) sequence of terms $\tau'_0, \ldots, \tau'_m, \tau'_j \neq \tau, 0 \leq j \leq m$, and such that $\tau \leq \tau'_0 \leq \cdots \leq \tau'_m \leq \tau_0$. Hence, our ontologies are directed trees rooted at τ_0 and whose edges capture generalisation relationships. Figure 1 illustrates an ontology rooted at the "*any*" term; term "*emergency*" is more general than "*ambulance*", and term "*any*" is more general than "*emergency*", that is, *ambulance* $\leq$ *emergency* $\leq$ *any*.

Next we define the subsumption relationship between the terms of a taxonomy.

DEF. 1. *For $\tau, \tau' \in \mathcal{T}$, we say that τ subsumes τ', denoted as $\tau' \sqsubseteq \tau$, iff there is a possibly empty sequence of terms $\tau'_0, \ldots, \tau'_m$ such that $\tau' \leq \tau'_0 \leq \cdots \leq \tau'_m \leq \tau$.*

In particular, we say that τ *strictly* subsumes τ', and we denote it as $\tau' \sqsubset \tau$, iff $\tau' \sqsubseteq \tau$ and $\tau' \neq \tau$. Term "*any*" subsumes terms "*any*", "*emergency*" and "*ambulance*", that is, *emergency* $\sqsubseteq$ *any* and *ambulance* $\sqsubseteq$ *ambulance*. We note that "*any*" *strictly* subsumes "*emergency*" and "*ambulance*", that is, *emergency* $\sqsubset$ *any*, *ambulance* $\sqsubset$ *any*.

Considering that the taxonomy of terms has a *tree* structure, the *intersection* between two terms is the most specific term subsuming these two terms. For instance, the intersection between terms "*ambulance*" and "*emergency*" is the term "*ambulance*". Formally:

DEF. 2. *For $\tau, \tau' \in \mathcal{T}$, their intersection $\tau \sqcap_t \tau'$ is:*

$$\tau \sqcap_t \tau' = \begin{cases} \tau & \text{if } \tau \sqsubseteq \tau' \\ \tau' & \text{if } \tau' \sqsubseteq \tau \\ \emptyset & \text{otherwise} \end{cases}$$

We denote by $\bar{\boldsymbol{\tau}}$ a vector of terms in $\mathcal{T}^n$, and we will refer to the i-th component of $\bar{\boldsymbol{\tau}}$ as τ_i. The intersection between two predicates $p(\bar{\boldsymbol{\tau}}), p(\bar{\boldsymbol{\tau}}') \in \mathcal{L}_{Ag}$ is another predicate with the intersection of each corresponding pair of terms in $\bar{\boldsymbol{\tau}}, \bar{\boldsymbol{\tau}}'$, whenever such intersection exists for all of them.

DEF. 3. *For $p(\bar{\boldsymbol{\tau}}), p(\bar{\boldsymbol{\tau}}') \in \mathcal{L}_{Ag}$, if $\tau_i \sqcap_t \tau'_i \neq \emptyset, 1 \leq i \leq n$, then their intersection $p(\bar{\boldsymbol{\tau}}) \sqcap_\pi p(\bar{\boldsymbol{\tau}}')$ is $p(\bar{\boldsymbol{\tau}}'')$ such that $\tau''_i = \tau_i \sqcap_t \tau'_i$ for all $1 \leq i \leq n$.*

In our running example (Figure 1) the intersection of *left*(*ambulance*) and *left*(*emergency*) is *left*(*ambulance*).

Taking inspiration from the anti-unification of terms proposed in [1], we define the *most specific generalisation* of terms. Given terms $\tau, \tau' \in \mathcal{T}, \tau \neq \tau'$, their *most specific generalisation* is the most specific term that *strictly* subsumes both of them. For instance in Figure 1, the most specific generalisation of "*ambulance*" and "*car*" is "*any*", since there is no other term which is more specific and strictly subsumes both of them. However, the most specific generalisation for "*ambulance*" and "*any*" does not exist because there is no term strictly subsuming "*any*". Formally:

DEF. 4. *For $\tau, \tau' \in \mathcal{T}, \tau \neq \tau'$, their* most specific generalisation*, denoted as $\tau \sqcup_t \tau'$, is a term $\tau_s \in \mathcal{T}$ such that $\tau \sqsubset \tau_s$ and $\tau' \sqsubset \tau_s$, and $\nexists \tau'' \in \mathcal{T}$ such that $\tau \sqsubset \tau'', \tau' \sqsubset \tau''$ and $\tau'' \sqsubset \tau_s$.*

We also define the most specific generalisation of predicates:

DEF. 5. *Predicates $p(\bar{\boldsymbol{\tau}}), p(\bar{\boldsymbol{\tau}}') \in \mathcal{L}_{Ag}$ have a* most specific generalisation *iff $\tau_i \sqcup_t \tau_i' \neq \emptyset, 1 \leq i \leq n$. Their most specific generalisation, denoted as $p(\bar{\boldsymbol{\tau}}) \sqcup_\pi p(\bar{\boldsymbol{\tau}}')$, is another predicate $p(\bar{\boldsymbol{\tau}}'')$ such that $\tau_i'' = \tau_i \sqcup_t \tau_i', 1 \leq i \leq n$.*

In our example, the most specific generalisation of *left*(*ambulance*), *left*(*car*) is *left*(*any*), with the most specific generalisation of terms "*ambulance*" and "*car*".

Finally, we define the generalisation relationship between norms:

DEF. 6. $\mathbf{n} = \langle \varphi, \theta(ac) \rangle$ *is more general than* $\mathbf{n}' = \langle \varphi', \theta(ac) \rangle$*, denoted as $\mathbf{n}' \subseteq \mathbf{n}$, iff $|\varphi| = |\varphi'|$, and for each predicate $p(\bar{\boldsymbol{\tau}}') \in \varphi'$, there is a predicate $p(\bar{\boldsymbol{\tau}}) \in \varphi, \tau_i' \sqsubseteq \tau_i, 1 \leq i \leq n$.*

We refer to $\mathbf{n}_1, \mathbf{n}_5$ from Section 2. Predicate *left*(*ambulance*) $\in \varphi_1$ (φ_i being the precondition of $\mathbf{n}_i$) has a corresponding predicate *left*(*emergency*) $\in \varphi_5$, *ambulance* $\sqsubseteq$ *emergency*. Similarly, *front*(*police*) $\in \varphi_1$ has *front*(*police*) $\in \varphi_5$, *police* $\sqsubseteq$ *police*, and *right*(*car*) $\in \varphi_1$ has *right*(*car*) $\in \varphi_5$, *car* $\sqsubseteq$ *car*. Norm $\mathbf{n}_5$ is thus more general than $\mathbf{n}_1$, $\mathbf{n}_1 \subseteq \mathbf{n}_5$.

3.2 Enhancing Norm Generalisation

We now describe how SIMON generalises norms. The process consists of: (i) monitoring when the norms of the normative system start performing well; (ii) checking if each identified norm is *generalisable* with the rest of norms; and (iii) generalising the norms if possible. Specifically, it first checks if the effectiveness and necessity performance values of a norm in the normative system surpass certain generalisation thresholds during the current time period. Second, for each norm that starts performing well, it checks if it is *generalisable* with another norm in the normative system. Notice that IRON considers all the other norms in the normative network (see Sect. 2) instead of just considering the ones on the normative system (i.e., the active norms). Third, in case two norms *are generalisable*, SIMON generalises them to their most specific general norm, namely their *parent* norm.

We define when two norms are generalisable:

DEF. 7. $\mathbf{n} = \langle \varphi, \theta(ac) \rangle$ *and* $\mathbf{n}' = \langle \varphi', \theta(ac) \rangle$, $\mathbf{n} \neq \mathbf{n}'$*, are generalisable iff for each predicate $p(\bar{\boldsymbol{\tau}}) \in \varphi$ either: (i) $p(\bar{\boldsymbol{\tau}}) \in \varphi'$; or (ii) there is a predicate $p(\bar{\boldsymbol{\tau}}') \in \varphi', p(\bar{\boldsymbol{\tau}}) \sqcup_\pi p(\bar{\boldsymbol{\tau}}') \neq \emptyset$.*

Figure 4 illustrates an example. It depicts a normative network at time t (Figure 4.1) and at time $t+1$ (Figure 4.2). At time t it contains two active norms $\mathbf{n}_1, \mathbf{n}_2$. Let us consider that both norms perform well enough to be generalised; both norms satisfy the conditions of Def. 7. Predicate *left*(*ambulance*) $\in \varphi_1$ has a corresponding predicate *left*(*police*) $\in \varphi_2$, *left*(*ambulance*) $\sqcup_\pi$ *left*(*police*) $=$ *left*(*emergency*) (i.e., their most specific generalisation is not empty). Predicate *front*(*police*) $\in \varphi_1$ has *front*(*police*) $\in \varphi_2$, and *right*(*car*) $\in \varphi_1$ has *right*(*car*) $\in \varphi_2$. Hence, $\mathbf{n}_1, \mathbf{n}_2$ are generalisable to a new norm that can be computed with Def. 8:

DEF. 8. *Given two generalisable (cf. Def 7) norms $\mathbf{n} = \langle \varphi, \theta(ac) \rangle, \mathbf{n}' = \langle \varphi', \theta(ac) \rangle$, their generalisation $\mathbf{n}'' = \langle \varphi'', \theta(ac) \rangle$ is such that for each predicate $p(\bar{\boldsymbol{\tau}}) \in \varphi$ and $p(\bar{\boldsymbol{\tau}}') \in \varphi'$, there is a predicate $p(\bar{\boldsymbol{\tau}}'') \in \varphi''$ obtained as:*

$$p(\bar{\boldsymbol{\tau}}'') = \begin{cases} p(\bar{\boldsymbol{\tau}}) & \text{if } \tau_i = \tau_i', 1 \leq i \leq n \\ p(\bar{\boldsymbol{\tau}}) \sqcup_\pi p(\bar{\boldsymbol{\tau}}') & \text{otherwise} \end{cases}$$

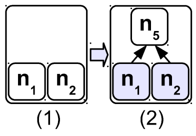

Figure 4: Direct generalisation of $\mathbf{n}_1, \mathbf{n}_2$.

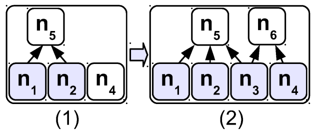

Figure 5: Indirect generalisation of $\mathbf{n}_4, \mathbf{n}_5$.

Norm $\mathbf{n}_5$ (Sec. 2) generalises $\mathbf{n}_1, \mathbf{n}_2$. It contains the most specific generalisation of predicates *left* of both $\mathbf{n}_1, \mathbf{n}_2$, and predicates *front* and *right* of $\mathbf{n}_1$. Once SIMON synthesises norm $\mathbf{n}_5$, it establishes generalisation relationships from $\mathbf{n}_1, \mathbf{n}_2$ to $\mathbf{n}_5$ as depicted in Figure 4.2. Hereafter we will refer to this generalisation as *direct* or *shallow* generalisation. Function $ShallowNormGeneralisation$ in Algorithm 1 implements it. Line 4 generalises $\mathbf{n}, \mathbf{n}'$ as $\mathbf{n}_p$, and line 6 establishes the corresponding generalisation relationships.

We note that SIMON generalises norms with *partial evidence*, while IRON requires *full evidence* to generalise norms. As an example, SIMON can synthesise $\mathbf{n}_5$ by generalising $\mathbf{n}_1, \mathbf{n}_2$, even though $\mathbf{n}_3$ has never been synthesised and no evidence has been gathered about its performance. In contrast, IRON will synthesise $\mathbf{n}_5$ only whenever $\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3$ have all been synthesised and there is evidence that they perform well. Therefore, we say that SIMON takes an *optimistic* approach to generalisation that requires less information than IRON to synthesise more compact normative systems.

Moreover, it is also worth noticing that IRON generalises norms taking into account all norms in the normative network whereas SIMON's norm generalisation only takes into account those norms in the normative system (that is, those that are active). Nevertheless, as Figure 6 shows, norms in the NS implicitly include other norms in the NN which in turn may be generalisable. Specifically, $\mathbf{n}, \mathbf{n}'$ in the normative system may not be directly generalisable, but they may include two other norms $\check{\mathbf{n}} \subseteq \mathbf{n}$, $\check{\mathbf{n}}' \subseteq \mathbf{n}'$ that will be generalisable if the conditions of Theorem 3.1 hold:

THEOREM 3.1. *Given $\mathbf{n} = \langle \varphi, \theta(ac) \rangle, \mathbf{n}' = \langle \varphi', \theta(ac) \rangle, \mathbf{n} \neq \mathbf{n}'$, satisfying the following conditions:*

1. *for each predicate $p(\bar{\boldsymbol{\tau}}) \in \varphi$ there is a predicate $p(\bar{\boldsymbol{\tau}}') \in \varphi'$ s.t. either $p(\bar{\boldsymbol{\tau}}) \sqcup_\pi p(\bar{\boldsymbol{\tau}}') \neq \emptyset$ or $p(\bar{\boldsymbol{\tau}}) \sqcap_\pi p(\bar{\boldsymbol{\tau}}') \neq \emptyset$; and*
2. *there is at least one pair of predicates $p(\bar{\boldsymbol{\tau}}) \in \varphi$ and $p(\bar{\boldsymbol{\tau}}') \in \varphi'$ such that $p(\bar{\boldsymbol{\tau}}) \sqcap_\pi p(\bar{\boldsymbol{\tau}}') = \emptyset$ and $p(\bar{\boldsymbol{\tau}}) \sqcup_\pi p(\bar{\boldsymbol{\tau}}') \neq \emptyset$.*

Then there exist two norms $\hat{\mathbf{n}} \subseteq \mathbf{n}$, $\check{\mathbf{n}}' \subseteq \mathbf{n}'$ that are generalisable.

PROOF. The proof proceeds by constructing norms $\check{\mathbf{n}}$, $\check{\mathbf{n}}'$ from $\mathbf{n}$ and $\mathbf{n}'$. We define $\check{\mathbf{n}}$, $\check{\mathbf{n}}'$ as the intersection of each pair of predicates $p(\bar{\boldsymbol{\tau}}) \in \varphi$ and $p(\bar{\boldsymbol{\tau}}') \in \varphi'$ as follows.

$$\check{\mathbf{n}} = \begin{cases} p(\bar{\boldsymbol{\tau}}) \sqcap_\pi p(\bar{\boldsymbol{\tau}}') & \text{if } p(\bar{\boldsymbol{\tau}}) \sqcap_\pi p(\bar{\boldsymbol{\tau}}') \neq \emptyset \\ p(\bar{\boldsymbol{\tau}}) & \text{otherwise} \end{cases}$$

$$\check{\mathbf{n}}' = \begin{cases} p(\bar{\boldsymbol{\tau}}) \sqcap_\pi p(\bar{\boldsymbol{\tau}}') & \text{if } p(\bar{\boldsymbol{\tau}}) \sqcap_\pi p(\bar{\boldsymbol{\tau}}') \neq \emptyset \\ p(\bar{\boldsymbol{\tau}}') & \text{otherwise} \end{cases}$$

We note that the method used to construct $\check{\mathbf{n}}, \check{\mathbf{n}}'$ guarantees that they share the predicates whose intersection is not empty. Therefore some of the predicates in both norms will be equal. When the intersection between two predicates is empty, we know from the assumptions in the theorem that there is a most specific generalisation for both predicates. In that case, we just copy the predicate of $\mathbf{n}$ into $\check{\mathbf{n}}$, and the predicate of $\mathbf{n}'$ into $\check{\mathbf{n}}'$. Therefore, by construction we ensure that for each pair or predicates in $\check{\mathbf{n}}, \check{\mathbf{n}}'$ either they are equal or there is a most specific generalisation for them. These are precisely the conditions that two norms must fulfil to be generalisable according to Definition 8. $\square$

As an example, consider the norms introduced in Section 2. SIMON adds $\mathbf{n}_4$ to the normative network represented in Fig. 4.2, yielding the normative network of Figure 5.1. We note that this normative network represents the normative system NS= $\{\mathbf{n}_4, \mathbf{n}_5\}$. Let us suppose $\mathbf{n}_4$ performs well, so SIMON assesses if it is generalisable with other norms in the normative system, namely $\mathbf{n}_5$. Following theorem 3.1 and the relationships between terms in Fig. 1, we see that: (i) for predicate *left*(*fire-brigade*) $\in \varphi_4$ there is a predicate *left*(*emergency*) $\in \varphi_5$, *left*(*fire-brigade*)$\sqcap_\pi$*left*(*emergency*) $=$ *left*(*fire-brigade*); (ii) for predicate *front*(*police*) $\in \varphi_4$ there is a predicate *front*(*police*) $\in \varphi_5$,*front*(*police*) $\sqcap_\pi$ *front*(*police*) $=$ *front*(*police*); and finally (iii) for predicate *right*(*police*) $\in \varphi_4$ there is a predicate *right*(*car*) $\in \varphi_5$, *right*(*car*) $\sqcap_\pi$ *right*(*police*) $=$ $\emptyset$ and *right*(*car*) $\sqcup_\pi$ *right*(*police*) $=$ *right*(*any*). Therefore, according to Def. 7, $\mathbf{n}_4$ and $\mathbf{n}_5$ are *indirectly* generalisable, and we can compute two norms $\check{\mathbf{n}} \subseteq \mathbf{n}_4$, $\check{\mathbf{n}}' \subseteq \mathbf{n}_5$ that are *directly* generalisable along the lines of the proof of theorem 3.1. Notice that in this case, $\check{\mathbf{n}} = \mathbf{n}_4$ and $\check{\mathbf{n}}' = \mathbf{n}_3$. Norm $\mathbf{n}_3$ contains (i) the intersection of *left*(*fire-brigade*) $\in \varphi_4$ and *left*(*emergency*) $\in \varphi_5$, (ii) the intersection of *front*(*police*) $\in \varphi_4$ and *front*(*police*) $\in \varphi_5$; and (ii) the most specific generalisation of *right*(*police*) $\in \varphi_4$ and *right*(*car*) $\in \varphi_5$. Finally, norms $\mathbf{n}_3, \mathbf{n}_5$ are generalised to $\mathbf{n}_6$.

Algorithm 1 details the complete norm generalisation process. Given a norm $\mathbf{n}$ that has become effective and necessary enough, function $DeepNormGeneralisation$ (line 9) carries out its generalisation with respect to a given normative network NN and an ontology $\mathcal{O}$. It starts by getting the active norms from the normative network NN and storing them in the normative system NS (line 10). Loop 11–17 then compares $\mathbf{n}$ with other norms $\mathbf{n}'$ in the normative system, computing norms $\check{\mathbf{n}} \subseteq \mathbf{n}, \check{\mathbf{n}}' \subseteq \mathbf{n}'$ that are generalisable, and the most specific norm $\mathbf{n}_p$ that generalises both of them (line 12). In lines 13–14 it checks if there exist norms $\check{\mathbf{n}}, \check{\mathbf{n}}', \mathbf{n}_p$, and norm $\mathbf{n}_p$ does not subsume any norm that underperforms (using $subsumesUnderperformingNorm$). If it is the case, then the algorithm works by first adding to NN those norms that do not yet belong to it (line 15) and then invoking function $addGenRelationship$ to update NN with new generalisation relationships stemming from those norms (line 16).

Function $getGeneralisableNorms$ (line 19) takes as parameters norms $\mathbf{n}, \mathbf{n}'$ and an ontology $\mathcal{O}$. Firstly, it computes the pair of norms $\check{\mathbf{n}} \subseteq \mathbf{n}$ and $\check{\mathbf{n}}' \subseteq \mathbf{n}'$ that are generalisable (line 20). If norms $\check{\mathbf{n}}$ and $\check{\mathbf{n}}'$ exist, then it creates the most specific norm that subsumes both of them, namely their parent norm $\mathbf{n}_p$ (line 23). Finally, it returns norms $\check{\mathbf{n}}$ and $\check{\mathbf{n}}'$, and the parent norm $\mathbf{n}_p$ (line 24).

3.3 Revising Over-Generalisations

As described in Section 3.2, SIMON requires partial evidence to generalise norms. As an example, consider the normative network depicted in Figure 4. SIMON generalises from norms $\mathbf{n}_1, \mathbf{n}_2$ to $\mathbf{n}_5$, which represents norms $\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3$, even though $\mathbf{n}_3$ has never been synthesised and there is no evidence on how it performs. This optimistic approach may lead to over-generalisations, since general norms may implicitly represent norms that have never been synthesised and may under-perform. For instance, let us suppose $\mathbf{n}_5$ does not perform well whenever agents apply it in the situation described by $\mathbf{n}_3$'s precondition. Therefore, SIMON has to specialise $\mathbf{n}_5$, activating $\mathbf{n}_1, \mathbf{n}_2$ and deactivating $\mathbf{n}_3$. However, in this case $\mathbf{n}_3$ does not exist in the normative network and hence cannot be deactivated, so it must be previously created in order to keep track of it and finally deactivate it. With this aim, in addition to norm specialisation, SIMON incorporates a method for refining norm generalisations whenever it detects that a general norm $\mathbf{n}$ under-performs in the situation described by a specific norm $\mathbf{n}_u$ that has not being created yet. This norm refining process is as follows: (i) it creates $\mathbf{n}_u$ in order to keep track of it, (ii) establishes a generalisation relationship from $\mathbf{n}_u$ to $\mathbf{n}$, and finally (iii) searches for alternative generalisation relationships that $\mathbf{n}_u$ may have with other norms in the normative network. We regard this process as a revision of generalisations, with the intention to *backtrack* the search for normative systems to a less general set of norms. Algorithm 3 shows how these steps come together in the overall strategy used in SIMON.

Algorithm 1 Shallow and Deep Norm Generalisation

```
 1: function SHALLOWNORMGENERALISATION($NN$, $\mathcal{O}$, $\mathbf{n}$)
 2:     $NS \leftarrow getNormativeSystem(NN)$
 3:     for all $\mathbf{n}' \in NS$ do
 4:         $\mathbf{n}_p \leftarrow mostSpecificGeneralisation(\mathbf{n}, \mathbf{n}', \mathcal{O})$
 5:         if $\mathbf{n}_p \neq null$ then
 6:             $NN \leftarrow addGenRelationship(NN, \{\mathbf{n}, \mathbf{n}'\}, \mathbf{n}_p)$
 7:     return $NN$
 8:
 9: function DEEPNORMGENERALISATION($NN$, $\mathcal{O}$, $\mathbf{n}$)
10:     $NS \leftarrow getNormativeSystem(NN)$
11:     for all $\mathbf{n}' \in NS$ do
12:         $(\check{\mathbf{n}}, \check{\mathbf{n}}', \mathbf{n}_p) \leftarrow getGeneralisableNorms(\mathbf{n}, \mathbf{n}', \mathcal{O})$
13:         if $\check{\mathbf{n}} \neq null$ and $\check{\mathbf{n}}' \neq null$
14:         and not $subsumesUnderperformingNorm(NN, \mathbf{n}_p)$ then
15:             $NN \leftarrow addToNNIfDoNotExist(\check{\mathbf{n}}, \check{\mathbf{n}}', \mathbf{n}_p)$
16:             $NN \leftarrow addGenRelationship(NN, \{\check{\mathbf{n}}, \check{\mathbf{n}}'\}, \mathbf{n}_p)$
17:     return $NN$
18:
19: function GETGENERALISABLENORMS($\mathbf{n}$, $\mathbf{n}'$, $\mathcal{O}$)
20:     $(\check{\mathbf{n}}, \check{\mathbf{n}}') \leftarrow getSubsumedGeneralisableNorms(\mathbf{n}, \mathbf{n}')$
21:     if $\check{\mathbf{n}} = null$ or $\check{\mathbf{n}}' = null$ then
22:         return $(null, null, null)$
23:     $\mathbf{n}_p \leftarrow mostSpecificGeneralisation(\check{\mathbf{n}}, \check{\mathbf{n}}', \mathcal{O})$
24:     return $(\check{\mathbf{n}}, \check{\mathbf{n}}', \mathbf{n}_p)$
```

Algorithm 2 Search relationships of a new norm $\mathbf{n}$.

```
 1: function SEARCHRELATIONSHIPS($NN$, $\mathcal{O}$, $\mathbf{n}$, $\mathbf{n}'$, $\mathbf{n}'_p$, $visited$)
 2:     if $\mathbf{n}' \subseteq_{\mathcal{O}} \mathbf{n}$ then
 3:         $NN \leftarrow addGenRelationship(NN, \{\mathbf{n}'\}, \mathbf{n})$
 4:         if $\mathbf{n} \subseteq_{\mathcal{O}} \mathbf{n}'_p$ then
 5:             $NN \leftarrow removeGenRelationship(NN, \mathbf{n}', \mathbf{n}'_p)$
 6:             $NN \leftarrow addGenRelationship(NN, \{\mathbf{n}\}, \mathbf{n}'_p)$
 7:     else
 8:         if $\mathbf{n} \subseteq_{\mathcal{O}} \mathbf{n}'$ then
 9:             $noChildSubsumesN \leftarrow True$
10:             $children \leftarrow getChildren(\mathbf{n}', NN)$
11:             while $noChildSubsumesN$ and $hasNext(children)$ do
12:                 $\mathbf{n}'_c \leftarrow getNext(children)$
13:                 if $\mathbf{n}'_c \notin visited$ then
14:                     $visited \leftarrow visited \cup \{\mathbf{n}'_c\}$
15:                     $NN \leftarrow searchRelationships(NN, \mathcal{O}, \mathbf{n}, \mathbf{n}'_c, \mathbf{n}', visited)$
16:                     if $\mathbf{n} \subseteq_{\mathcal{O}} \mathbf{n}'_c$ then
17:                         $noChildSubsumesN \leftarrow False$
18:             end while
19:             if $noChildSubsumesN$ then
20:                 $NN \leftarrow addGenRelationship(NN, \{\mathbf{n}\}, \mathbf{n}')$
21:     return $NN$
```

This last step in the refinement method is an important stage in the overall SIMON strategy. It concerns searching for alternative generalisation relationships a norm $\mathbf{n}$ may have with other norms $\mathbf{n}'$ in the normative network. Algorithm 2 details this stage. First, if $\mathbf{n}$ is more general than $\mathbf{n}'$ with respect to ontology $\mathcal{O}$ (denoted as $\mathbf{n}' \subseteq_{\mathcal{O}} \mathbf{n}$), then $\mathbf{n}$ is a parent of $\mathbf{n}'$ (line 2). Hence, it establishes a generalisation relationship[1] from $\mathbf{n}'$ to $\mathbf{n}$ (line 3), as depicted in Figure 7.1. Additionally, if any parent norm $\mathbf{n}'_p$ of $\mathbf{n}'$ is more general than $\mathbf{n}$ (line 4), it means that $\mathbf{n}$ should be inserted between $\mathbf{n}'$ and its parent $\mathbf{n}'_p$. Therefore, it removes the generalisation relation-

[1] A generalisation relationship is established between two norms by invoking function $addGenRelationship()$, which creates an edge between them in the NN iff it does not yet exist.

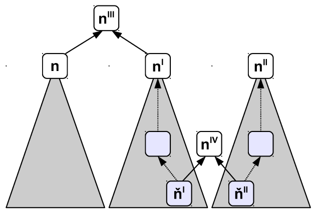

Figure 6: Direct generalisation of norms $\mathbf{n}$ and $\mathbf{n}'$ and indirect generalisation of norms $\mathbf{n}'$ and $\mathbf{n}''$.

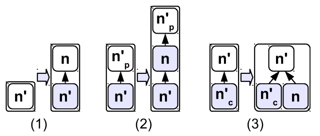

Figure 7: Examples of establishing generalisation relationships between a new norm $\mathbf{n}$ and a norm $\mathbf{n}'$ in the NN.

ship from $\mathbf{n}'$ to its parent $\mathbf{n}'_p$ (line 5), and establishes a generalisation relationship from $\mathbf{n}$ to $\mathbf{n}'_p$ (line 6). Figure 7.2 illustrates this operation. Second, it checks if $\mathbf{n}'$ subsumes $\mathbf{n}$ (line 8). If this is so, it may be the case that $\mathbf{n}$ is further subsumed by a children $\mathbf{n}'_c$ of $\mathbf{n}'$ or, alternatively, that $\mathbf{n}$ should be inserted as a sibling of the children of $\mathbf{n}'$ because none of them subsumes $\mathbf{n}$. The former is checked in lines 9–18 whereas the later (which establishes a generalisation relationship from $\mathbf{n}$ to $\mathbf{n}'$ as illustrated in Figure 7.3) is implemented in lines 19–20. As for the former alternative, the algorithm initialises the Boolean variable *noChildSubsumesN* to *True* (line 9) as well as *children* to the list of children of norm $\mathbf{n}'$ (line 10). Then, it loops through all the children $\mathbf{n}'_c$ until a child is found to subsume $\mathbf{n}$ (lines 11–12 and 16–17). For each considered (and not visited) child, there is a recursive invocation of the function so that it searches for relationships between $\mathbf{n}$ and that children (lines 13-15).

3.4 A New Strategy for Norm Synthesis

We have so far described a new approach to perform and to revise/backtrack norm generalisations. Next, we introduce a novel strategy for norm synthesis, using optimistic norm generalisation. Since SIMON is an on-line method, its norm synthesis strategy is continuously executed. SIMON performs conflict detection and synthesises new norms as described in Section 2, except that SIMON does not generate potential norm generalisations. Crucially, the norm evaluation and norm refinement phases are novel. In addition to evaluating norms, the norm evaluation phase synthesises under-performing norms that have never been synthesised but are implicitly represented by general norms. Finally, norm refinement generalises norms taking the optimistic generalisation approach described in Sect. 3.2, and specialises norms as described in Sect. 2.

Algorithm 3 describes the new strategy, where, for each detected conflict (line 2) it creates new norms (line 3) aimed at avoiding conflicts in the future. Then, norm evaluation evaluates applicable norms (lines 4–5) and returns norm performances P and a set of *negatively rewarded norms* (NRN), namely those under-performing norms that do not exist but are implicitly represented by general norms. Next, it carries out norm refinement. First, it adds to the normative network each norm in NRN (lines 7–8), and searches for their possible relationships with other norms in the NN (lines 9–10). Second, it performs the optimistic generalisation of norms described in Section 3.2 whenever it detects that they start performing well (lines 13-17). The *mode* parameter determines whether to invoke our *Shallow* or *Deep* generalisation methods. Third, it specialises norms whenever it detects they have just become ineffective or unnecessary (lines 18–19).

3.5 Evaluating Normative Systems

In addition to the effectiveness and necessity measures introduced in IRON, we provide two further metrics introduced by [4], namely minimality and simplicity. Minimality is concerned with minimising the amount of constraints (in a normative system) imposed on

Algorithm 3 SIMON's norm synthesis strategy

```
1:  function SIMONSTRATEGY($views, NN, \mathcal{O}, mode, step$)
2:      $conflicts \leftarrow conflictDetection(views)$
3:      $(createdNorms, NN) = normCreation(conflicts, NN, \mathcal{O})$
4:      $applicableNorms \leftarrow normApplicability(views, NN)$
5:      $(P, NRN) \leftarrow normEvaluation(applicableNorms)$
6:      for all $\mathbf{n} \in NRN$ do
7:          $NN \leftarrow add(NN, \mathbf{n})$
8:          $NS \leftarrow getNormativeSystem(NN)$
9:          for all $\mathbf{n}' \in NS$ do
10:             $NN \leftarrow searchRelationships(NN, \mathcal{O}, \mathbf{n}, \mathbf{n}', null, \emptyset)$
11:     for all $\mathbf{n} \in applicableNorms$ do
12:         if $crossedGeneralisationThreshold(\mathbf{n}, P)$ then
13:             if $mode$ = S-SIMON then
14:                 $NN \leftarrow ShallowNormGeneralisation(NN, \mathcal{O}, \mathbf{n})$
15:             else
16:                 if $mode$ = D-SIMON then
17:                     $NN \leftarrow DeepNormGeneralisation(NN, \mathcal{O}, \mathbf{n})$
18:         if $crossedSpecialisationThreshold(\mathbf{n}, P)$ then
19:             $NN \leftarrow normSpecialisation(NN, \mathbf{n}, P)$
20:     return $NN$
```

agents. The more minimal a normative system, the greater the individual agent freedom. Simplicity refers to norms that are easy to reason about by agents. The simpler the norms, the less computational resources required to reason about them. We note, therefore, that minimality and simplicity are local (agent-level) synthesis criteria, aimed at simplifying the reasoning of individual agents. We naturally capture both concepts by measuring the size of a normative system (minimality) and its number of clauses (simplicity):

DEF. 9. *The minimality of NS is* $\mathcal{M}(NS) = |NS|$.

DEF. 10. *The simplicity of NS is* $\mathcal{S}(NS) = \sum_{\langle\varphi,\theta(ac)\rangle \in NS} |\varphi|$.

These two measures are key to the problem at hand (i.e., the on-line synthesis of normative systems) since the smaller the minimality and simplicity of normative systems, the better it is so as to give agents flexibility, to save the agents' computational resources (when processing norms), and to avoid over-regulation.

4. EMPIRICAL EVALUATION

We now compare the performance of SIMON with IRON along several dimensions. First, we compare both approaches in terms of the quality (of minimality, simplicity, effectiveness and necessity) of the normative systems they synthesise, as well as the convergence time required by their synthesis. We then perform a micro analysis of the distributions of normative systems synthesised by IRON and SIMON. Our purpose is to shed light on how the different generalisation mechanisms employed by IRON and SIMON affect their norm synthesis processes. This will help us understand the differences in quality of the normative systems that they synthesise. Finally, we also perform an analysis of the computational costs of both approaches. In our empirical analysis we employed the implementation of IRON made publicly available by the authors.[2]

4.1 Empirical settings

We recall from Section 3 that SIMON offers two operation modes, namely *Shallow Simon* (S-SIMON) and *Deep Simon* (D-SIMON). Furthermore, both versions of SIMON employ a generalisation step. In our experiments we will employ both S-SIMON and D-SIMON in our comparison with IRON.

Our experiments use the same scenario and experimental settings described in [7]. We run a discrete-event simulation of a traffic junction, with agents being autonomous cars, and conflicts

[2] IRON: http://www.iiia.csic.es/∼jmorales/Downloads/IRON.zip

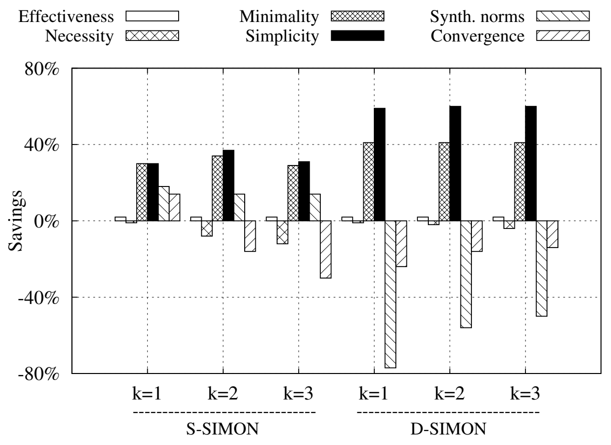

Figure 8: Savings of S-SIMON and D-SIMON with respect to IRON.

their collisions. Each simulation incorporates some norm synthesis mechanism. At simulation time, every time the synthesis mechanism in use changes the current normative system, it sends the new normative system to the cars. At each tick, each car decides whether to apply or violate the current normative system's norms. This decision is made according to some violation probability, which is fixed to 0.3 and is the same for all cars.

We have performed 200 simulations for each norm synthesis mechanism, namely for IRON, S-SIMON, and D-SIMON. Following [7]: we start with an empty normative system; norms are created with 0.5 initial effectiveness/necessity; and norms have at most 3 predicates (clauses). An important parameter in this comparison is what we call the *generalisation step*. Notice that, from definition 8, a generalisation can involve from 1 clause up to the total number of clauses in the precondition of a norm. Thus, a generalisation can take up to at most n clauses at the same time. Our simulations using S-SIMON and D-SIMON consider values for the generalisation step, k, within [1..3]. For instance, setting $k = 2$ means that a generalisation can involve both 1 and 2 clauses. Observe that unlike S-SIMON and D-SIMON IRON can only perform generalisations of a single predicate (which is equivalent to fixing the generalisation step to 1). Each simulation concludes, hence converging to a normative system if during a 4000-tick period: (i) the normative system remains stable; and (ii) no new conflicts (those that have not been regulated by any norm yet) arise. Thus, conflicts arising from norm violations are disregarded when assessing convergence. Our results below show the *interquartile mean* in the third quartile of the different measures employed in our comparison.

	IRON	S-SIMON			D-SIMON		
		$k=1$	$k=2$	$k=3$	$k=1$	$k=2$	$k=3$
Effectiveness	0.853	2.6%	2.32%	2.68%	2.66%	2.29%	2.6%
Necessity	0.487	-1%	-8.1%	-12.9%	-1.7%	-2.1%	-4.1%
Minimality	8.76	30.67%	34.47%	29.52%	41.01%	41.85%	41.93%
Simplicity	18.313	30.14%	37.2%	31.19%	59.19%	60.1%	60.68%
Synth. norms	23.5	18.55%	14.38%	14.38%	-77%	-56.82%	-50.41%
Norm accesses	86,256	99.8%	99.69%	99.57%	99.29%	99.3%	99.27%
Convergence	6,323.6	14.79%	-16.56%	-30.51%	-24.68%	-16.01%	-14.18%

Table 1: Savings values of S-SIMON and D-SIMON w.r.t. IRON.

4.2 Empirical results

Our first comparison mainly focuses on the quality of the normative systems obtained by S-SIMON, D-SIMON, and IRON. Figure 8 illustrates the savings of S-SIMON and D-SIMON with respect to IRON as the generalisation step k increases, for $k \in \{1, 2, 3\}$. Additionally, Table 1 shows numerical data corresponding to Fig. 8.

We initially analyse the quality obtained by the three algorithms being compared. Notice that S-SIMON with $k = 1$ already synthesises normative systems that are up to 30.67% more minimal and 30.14% simpler than those synthesised by IRON, while keeping effectiveness and necessity at very similar values. Here S-SIMON clearly benefits from its more optimistic norm generalisation, which allows it to generalise further than IRON, synthesising more compact normative systems. When increasing the generalisation step, $k = 2$, S-SIMON obtains further benefits in terms of minimality and simplicity (34.47% and 37.2% respectively). Nonetheless, increasing the generalisation step, $k = 3$, makes S-SIMON obtain worse results than for $k = 2$. Therefore, S-SIMON cannot take advantage of the largest generalisation step. Recall that a large generalisation step enables S-SIMON to eventually carry out large generalisations. However, this may lead to over-generalisations. Then, we have observed that after undoing an over-generalisation, S-SIMON ends up with a normative network where it finds difficult to perform generalisations. A norm representing an over-generalisation causes the synthesis in the normative network of child norms that are negatively rewarded. When backtracking an over-generalisation S-SIMON ends up with a larger number of norms (than before generalising) that it finds difficult to generalise.

Figure 8 also shows that D-SIMON clearly outperforms both IRON and S-SIMON, thus being the best in class. When using a low generalisation step, $k = 1$, D-SIMON already synthesises normative systems that are up to 41.01% more minimal and 59.19% simpler than those synthesised by IRON, while keeping effectiveness and necessity at very similar values. D-SIMON also outperforms S-SIMON, obtaining benefits in terms of minimality and simplicity, namely the normative systems synthesised by D-SIMON have fewer norms and fewer clauses than those synthesised by S-SIMON. Notice also that D-SIMON slightly benefits in terms of minimality and simplicity as the generalisation step increases. That means that in this particular domain, $k = 1$ is enough for D-SIMON to obtain very compact normative systems.

When we consider the convergence time required by S-SIMON, D-SIMON, and IRON, we observe that as the generalisation step increases, S-SIMON requires more time to converge, whereas D-SIMON requires less time to converge (in terms of the number of ticks). This indicates that increasing the generalisation step helps D-SIMON reach convergence. In other words, the generalisation steps favours D-SIMON's synthesis process. This is not the case for S-SIMON. As discussed above the largest value of the generalisation step is detrimental to S-SIMON, leading to extra work to undo over-generalisations.

To summarise D-SIMON with k=3 is the best-in-class algorithm since it achieves the best (lowest) values of minimality and simplicity at a low cost of extra convergence time with respect to IRON.

Micro analysis. We now investigate why D-SIMON and S-SIMON both outperform IRON in terms of minimality and simplicity. Figure 9 shows three histograms of the normative systems synthesised by D-SIMON, S-SIMON and IRON. These histograms consider D-SIMON with $k = 3$ and S-SIMON with $k = 2$, namely the best D-SIMON and S-SIMON algorithms. The x-axis in each histogram shows the different normative systems synthesised by the three algorithms, while the y-axis shows the number of times each normative system was synthesised. Overall, the three algorithms together managed to synthesise 314 *different* normative systems.

We first observe in Figure 9c that IRON synthesises 173 different normative systems, ranging from NS_{142} to NS_{314}. When we consider Figure 9b depicting S-SIMON's histogram, we observe that

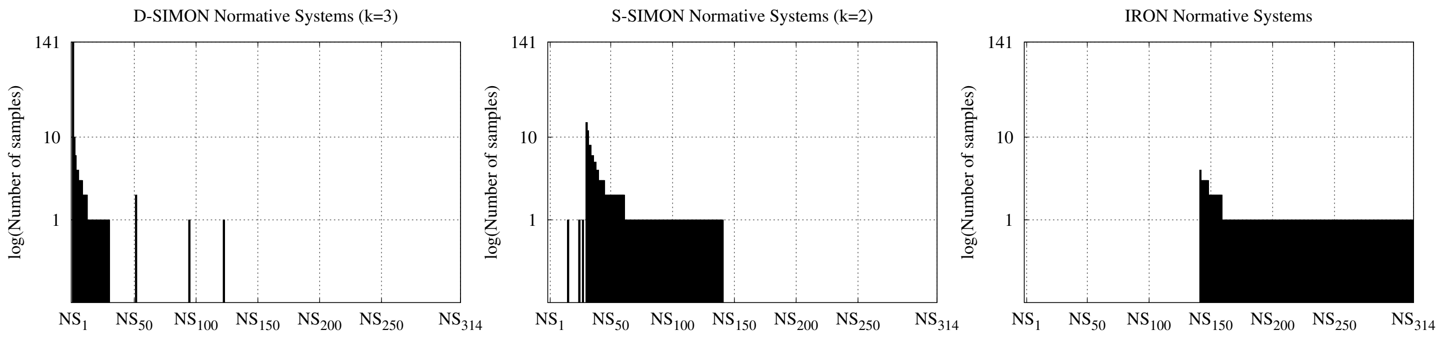

Figure 9: From left to right, histograms of the different normative systems synthesised by: a) D-SIMON; b) S-SIMON; c) IRON.

S-SIMON synthesises 111 normative systems, mostly ranging from NS_{31} to NS_{141}. Therefore, S-SIMON explores an area of the space of normative systems that is not at all addressed by IRON. Finally, when we consider Figure 9a depicting D-SIMON's histogram, we observe that D-SIMON synthesises 30 different normative systems, mostly ranging from NS_1 to NS_{30}. Although there is some intersection between the normative systems synthesised by D-SIMON and S-SIMON, we notice that, again, D-SIMON mostly explores an area of the space of normative systems that is not reached by S-SIMON or IRON. To summarise, the three algorithms explore different areas of the search space of normative systems.

Next we analyse the dispersion of the three distributions of synthesised normative systems. Notice that D-SIMON's dispersion is low (it only synthesises 30 different normative systems out of 200 different simulations), whereas S-SIMON's is medium (it synthesises 111 normative systems), and IRON's is high (it synthesises 173 different normative system). In fact, D-SIMON synthesises 141 times (70.5%) normative system NS_1, and the first 10 normative systems are synthesised 89% of the times. Against this, S-SIMON synthesises normative system NS_{31} only 15 times (7.5% of the total), and IRON synthesises NS_{142} only 4 times (2% of the total).

To summarise, D-SIMON consistently focuses on an area of the search space where more minimal and simpler normative systems are. This explains why it outperforms S-SIMON and IRON.

Computational cost analysis. Finally, we compare D-SIMON, S-SIMON, and IRON with respect to the number of norm accesses in the normative network and the number of synthesised norms[3]. The results are shown in the table of Fig. 1. On the one hand, both D-SIMON and S-SIMON require a significantly lower number of norm accesses than IRON to synthesise norms (in both cases over 99% fewer accesses). On the other hand, while S-SIMON synthesises fewer norms than IRON (between 14.38% and 18.55%, depending on the generalisation step), D-SIMON does synthesise more norms than IRON (between 50.41% and 77%, depending on the generalisation step). This must not be regarded negatively because this is precisely what allows D-SIMON to synthesise normative systems that are reached by neither S-SIMON nor IRON. In fact, D-SIMON synthesises more norms that its competitors because it manages to perform more norm generalisations than them, which amounts to synthesising new (general) norms.

5. CONCLUSIONS & FUTURE WORK

We have presented SIMON, a mechanism for the on-line synthesis of norms, which outperforms IRON [7], a recently proposed approach. We structure domain knowledge as a tree representing a taxonomy, and compute the *most specific generalisation* of terms to select alternative (more general) terms for our norms; we provide a means to handle over-generalisation, whereby SIMON backtracks to more specific norms, increasing precision. To our knowledge, our work is the first to experimentally tackle *compactness* issues (minimality and simplicity) in normative systems. We provided empirical evaluation of the quality our approach, compared with IRON [7], also exploring alternative operation modes (namely, deep and shallow). SIMON's exploration of the search space of norms is more efficient, and the synthesised normative systems have fewer and simpler norms. SIMON offers a fine-grained control of how to generalise the pre-condition of norms – up to n predicates can be generalised at the same time. We are currently exploring our approach in different domains, namely, self-regulation of on-line communities, as well as best-practices for professional bodies.

[3] One should not confuse the number of synthesised norms with the number of normative systems.

6. REFERENCES

[1] E. Armengol and E. Plaza. Bottom-up induction of feature terms. *Machine Learning*, 41(3):259–294, 2000.

[2] G. Boella, L. van der Torre, and H. Verhagen. Introduction to normative multiagent systems. *Computational & Mathematical Organization Theory*, 12(2-3):71–79, 2006.

[3] F. Dignum. Autonomous agents with norms. *Artif. Intell. Law*, 7(1):69–79, 1999.

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

[5] N. Griffiths and M. Luck. Norm Emergence in Tag-Based Cooperation. In *Proceedings of COIN*, 2010.

[6] J. Morales, M. López-Sánchez, and M. Esteva. Using Experience to Generate New Regulations. In *IJCAI*, pages 307–312, 2011.

[7] J. Morales, M. Lopez-Sanchez, J. A. Rodriguez-Aguilar, M. Wooldridge, and W. Vasconcelos. Automated synthesis of normative systems. In *AAMAS 2013*, pages 483–490, 2013.

[8] O. Sen and S. Sen. Effects of social network topology and options on norm emergence. In *Proceedings of COIN*, pages 211–222, 2010.

[9] S. Sen and S. Airiau. Emergence of norms through social learning. In *IJCAI*, pages 1507–1512, 2007.

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

[11] D. Villatoro, J. Sabater-Mir, and S. Sen. Social instruments for robust convention emergence. In *IJCAI*, pages 420–425, 2011.