# Offline Evolution of Normative Systems

Magnus Hjelmblom[1,2]
[1]*Faculty of Engineering and Sustainable Development, University of Gävle, SE-80176 Gävle, Sweden*
[2]*Department of Computer and Systems Sciences, Stockholm University, Forum 100, SE-16440, Kista, Sweden*

Keywords: Norm-regulated Multi-Agent System, Normative MAS, DALMAS, Norm Evolution, Evolutionary Algorithm.

Abstract: An approach to the pre-runtime design of normative systems for problem-solving multi-agent systems (MAS) is suggested. A key element of this approach is to employ evolutionary mechanisms to evolve efficient normative systems. To illustrate, a genetic algoritm is used in the process of designing a normative system for an example MAS based on the DALMAS architecture for norm-regulated MAS. It is demonstrated that an evolutionary algorithm may be a useful tool when designing norms for problem-solving MAS.

## 1 INTRODUCTION

Agent-based modeling and simulation is an active field of study which, for example, may offer methods for solving complex optimisation problems. In this setting, agents are required to cooperate to solve the problem at hand. In complex systems with adjustable agent autonomy, sophisticated planning can often be replaced by norms; see for example (Verhagen and Boman, 1999). The study of norm-regulated multi-agent systems, often referred to as normative MAS, has also attracted a lot of attention. The NorMAS roadmap (Andrighetto et al., 2013b) is a comprehensive introduction to and overview of the field. The combination of agent-based modeling and simulation and normative MAS is a promising field of study. (Balke et al., 2013)

It is often desirable to replace planning (and replanning), since it may be a complex and time-consuming task, especially in collaborative environments. On the other hand, designing good normative systems is also a challenge. The approach suggested here, whose basic ideas were outlined in (Odelstad and Boman, 2004, pp. 164f), is to use evolutionary mechanisms, employed in a genetic algorithm, to aid the 'off-line' (i.e., pre-runtime) design of normative systems for problem-solving multi-agent systems.

The paper is structured as follows. In Sect. 1.2, previous work on the DALMAS architecture for norm-regulated MAS is briefly presented. Sect. 2 introduces an example DALMAS which is used in Sect. 3 to demonstrate how to employ evolutionary mechanisms in the process of designing norms, by applying an evolutionary algorithm to this example. Sect. 4 concludes and suggests some lines of future work.

### 1.1 Related Work

The runtime emergence of norms within artificial social systems has attracted the attention of many researchers; see, e.g., (Andrighetto et al., 2013a). However, evolving normative systems as part of the process of designing norm-regulated MAS is not as well studied, but evolutionary approaches for learning behaviour patterns or strategies for coordination have been successfully used in, e.g., the RoboCup[1] domain; see for example (Luke et al., 1998; Di Pietro et al., 2002; Nakashima et al., 2004). In fact, the simple decision policies evolved by Di Pietro et al. for the RoboCup *Keepaway* game can be regarded as simple normative systems consisting of production rules which prescribe certain behaviours in certain situations.

### 1.2 Previous Work

DALMAS (Odelstad and Boman, 2004) is an abstract architecture for a class of norm-regulated multi-agent systems. A deterministic DALMAS is a simple multi-agent system in which the actions of an agent are connected to transitions between system states. In a deterministic DALMAS the agents take turns to act; only one agent at a time may perform an action. By allowing 'do nothing' actions and accelerating the turn-taking, systems with close to asynchronous behaviour can be obtained.

Formally, a DALMAS is an ordered 9-tuple, where the components are various sets, operators and functions which give the specific DALMAS its unique fea-

[1]http://www.robocup.org

Hjelmblom M..
Offline Evolution of Normative Systems.
DOI: 10.5220/0005284102130221
In *Proceedings of the International Conference on Agents and Artificial Intelligence* (ICAART-2015), pages 213-221
ISBN: 978-989-758-074-1

tures. Of particular interest is the deontic structure-operator, which for each situation of the system determines an agent's deontic structure (i.e., the set of permissible acts) on the feasible acts in the current situation, and the preference structure-operator, which for each situation determines the preference structure on the permissible acts. A norm-regulated simple deterministic DALMAS employs what is often referred to as 'negative permission', by letting the deontic structure consist of all acts that are not explicitly prohibited by a normative system. The preference structure consists of the most preferable (according to the agent's utility function) of the acts in the deontic structure. In short, a DALMAS agent's behaviour is regulated by the combination of a normative system and a utility function; this 'agent oeconomicus norma'[2] chooses the most desirable act, according to the utility function, within the 'room for manouver' determined by the norms. The DALMAS's normative framework is based on an algebraic version of the Kanger-Lindahl theory of normative positions, in which normative consequences are formulated by applying normative operators to descriptive conditions. From these general normative sentences on conditions follow normative sentences regarding specific states of affairs, which in turn result in permission or prohibition of individual actions in specific situations. (See for example (Lindahl, 1977; Lindahl and Odelstad, 2004; Odelstad and Boman, 2004; Odelstad, 2008) for an introduction.) Hence, the norms in the DALMAS architecture play a different role, and is represented in a fundamentally different way, than, e.g., the decision rules in the *RoboCup* setting (see Sect. 1.1).

Since the agents take turns to act, each individual step in a run of a DALMAS may be characterised by an ordered 5-tuple $S = \langle x, s, A, \Omega, S \rangle$ whose components are a set of states $S$, a state $s$, an agent-set $\Omega = \{x_1, ..., x_n\}$, the acting ('moving') agent $x$, and an action-set $A = \{a_1, ..., a_m\}$.[3] In this setting, $a$ may be regarded as a function such that $a(x,s) = s^+$ means that $s^+$ is the resulting state when $x$ performs act $a$ in state $s$. In the following, the abbreviation $s^+$ will be used for $a(x,s)$ when there is no need for an explicit reference to the action $a$ and the acting agent $x$. Since the action by the acting agent is deterministic and is performed asynchronously, there is no simultaneous action by other agents (including the 'environment', which may be regarded as a special kind of agent). Furthermore, we assume that a $\nu$-ary condition $d$ is true or false on $\nu$ agents $x_1, ..., x_\nu \in \Omega$ in $s$; this will be written $d(x_1, ..., x_\nu; s)$.[4] To facilitate the presentation, $X_\nu$ will often be used as an abbreviation for the argument sequence $x_1, ..., x_\nu$. Negations $d'$, conjunctions $(c \wedge d)$ and disjunctions $(c \vee d)$ can be formed in the following way:

$d'(X_\nu)$ iff $\neg d(X_\nu)$,
$(c \wedge d)(X_\nu)$ iff $c(X_p)$ and $d(X_q)$, and
$(c \vee d)(X_\nu)$ iff $c(X_p)$ or $d(X_q)$

where $\nu = \max(p,q)$.[5] Therefore, it is possible to construct Boolean algebras of conditions.

Let the situation $\langle x, s \rangle$ be characterised by the moving agent $x$ and the state $s$ in a norm-regulated simple deterministic DALMAS. In the following, we assume that norms always apply to the moving agent $x$ in a situation $\langle x, s \rangle$, in order to facilitate the presentation. A norm in $N$ is represented by an ordered pair $\langle g, c \rangle$, where the (descriptive) condition $g$ on a situation $\langle x, s \rangle$ is the ground of the norm and the (normative) condition $c$ on $\langle x, s \rangle$ is its consequence; see, e.g., (Odelstad and Boman, 2004). We first define a set of 'transition type operators' $C_k^a$, based on Table 2 in (Hjelmblom, 2014a), and a set of corresponding 'transition type prohibition operators' $P_k$, $k \in \{1, 2\Lambda, 2\Omega, 4\Lambda, 4\Omega, 5, 6\Lambda, 6\Omega, 7\}$, such that $P_k d(X_\nu; x, s)$ is intended to mean that if $C_k^a d(X_\nu; x, s)$ holds, then $a$ is prohibited for $x$ in $\langle x, s \rangle$.[6] In effect, $P_k d(X_\nu; x, s)$ implies a prohibition of zero, one or two of the four 'basic transition types' with regard to the state of affairs $d(X_\nu)$.[7] For example, $\langle c, P_k d \rangle$, where $c$ and $d$ can have different arity, represents the sentence

$$\forall x_1, x_2, ..., x_\nu \in \Omega : c(x_1, x_2, ..., x_p; x, s) \rightarrow P_k d(x_1, x_2, ..., x_q; x, s)$$

where $\Omega$ is the set of agents, $x$ is the acting agent (to which the norm applies) in the situation $\langle x, s \rangle$, and $\nu = \max(p,q)$. If the condition specified by the ground of a norm for some agents in some situation, then the (normative) consequence of the norm is in effect in that situation. If the normative system contains a norm whose ground holds in the situation $\langle x, s \rangle$

[2] Cf. (Odelstad, 2008, Sect. 1.8.3).

[3] In (Hjelmblom, 2013) such a tuple is called a *transition system situation*.

[4] In the special case when the sequence of agents is empty, i.e. $\nu = 0$, $d$ represents a proposition which is true or false in $s$.

[5] The free variables in $c(x_1, ..., x_p)$ must be the same, and in the same order, as the free variables in $d(x_1, ..., x_q)$, but it is not necessary that $p$ and $q$ have the same arity. Cf. (Odelstad and Boman, 2004, p. 146).

[6] The original set of operators in (Odelstad and Boman, 2004) contains seven operators, indexed 1-7. In (Hjelmblom, 2013), two new operators were added. See this paper for an explanation of the somewhat peculiar indices.

[7] See (Hjelmblom, 2014a, Sect. 2) for a description of the basic transition types.

and whose consequence prohibits the type of transition represented by $x$ performing action $a$, then $a$ is prohibited for $x$ in $\langle x,s\rangle$:

$$\begin{gathered}Prohibited_{x,s}(a) \text{ according to } N\\ \text{if there exists a } p\text{-ary condition } c\\ \text{and a } q\text{-ary condition } d\\ \text{and a } k \in \{1,2\Lambda,2\Omega,4\Lambda,4\Omega,5,6\Lambda,6\Omega,7\},\\ \text{such that } \langle c,P_k d\rangle \text{ is a norm in } N,\\ \text{and there exist } x_1,...,x_\nu \text{ such that}\\ c(x_1,...,x_p;x,s)\ \&\ C_k^a d(x_1,...,x_q;x,s),\\ \text{where } \nu = \max(p,q).\end{gathered}$$

Hence, if $c(x_1,...,x_p;x,s)$ for some sequence of agents $x_1,...,x_\nu$, then the normative condition $P_k d(x_1,...,x_q;x,s)$ is 'in effect'. Thus, if $C_k^a d(x_1,...,x_q;x,s)$ holds, then $a$ is prohibited for $x$ in $s$. (Cf. the examples in Sect. 3.1.) Table 1 shows the set of nine norm-building operators, together with (in the rightmost column) the corresponding $C_k^a$ applied to $d(x_1,...,x_q;s)$. Cf. Table VI in (Hjelmblom, 2014b), which also shows a suggested interpretation of the $P_k$ operators in terms of an extended set of types of one-agent normative positions, based on the Kanger-Lindahl theory of normative positions.

A general-level Java/Prolog implementation of the DALMAS architecture has been developed, to facilitate the implementation of specific systems. The Colour & Form system, the Waste-collector system and the Forest Cleaner system are three specific systems that have been implemented using this framework. The reader is referred to (Odelstad and Boman, 2004; Hjelmblom, 2008; Hjelmblom and Odelstad, 2009; Hjelmblom, 2011) for a description of these systems and their implementations.

The approach to normative systems employed in this framework is ideally suited for evolution of normative systems, since the set of $P_k$ operators exhausts the set of logical possibilities regarding prohibition of transition types. Therefore each conceivable normative system, consisting of conditional norms based on descriptive conditions selected from a set of potential grounds and normative conditions selected from a set of potential consequences, could become a candidate for evaluation in the execution of an evolutionary algorithm. This idea will be further explored in the following sections.

## 2 EXAMPLE: EXPLORER DALMAS

Let us consider a class of systems of agents operating in an environment consisting of a grid of squares ordered in rows and columns, in which each square is assigned a pair of integer coordinates. Let us assume that the joint goal of the agents is to explore as much as possible of the grid using a fixed number of moves. An agent can stay in the current square, i.e., do nothing, or move one square in one of eight directions (east, northeast, north, northwest, west, southwest, south, southeast) as long as it stays within the boundaries of the grid. In other words, in a given situation, an action is feasible if and only if it does not move the agent off limits. It should of course be noted that these simple systems (in the following referred to as Explorer DALMASes) in themselves are of limited interest, but the idea here is to illustrate how evolutionary mechanisms could be used in the process of designing normative systems for problem-solving MAS.

To simulate a situation with limited possibilites for communication between agents and only local knowledge of the environment, we further assume that an agent only knows the status (visited or unvisited) of the immediately surrounding squares, and the location of other agents within two squares. An agent's preference is represented by a very simple utility function such that moving to an unvisited square is preferred over moving to a visited square, and stay is the least preferred action. In the case of a tie between equally preferred actions, one of them is randomly selected. In other words, all agents have the same utility function.

To make the situation more concrete, let us assume that the size of the grid is $7\times 7$ squares and place three agents at square $(1,1)$, the leftmost lowest square. Note that this system can be considered as an instance of the Waste-collector system, in which visited (resp., unvisited) squares are represented by 0 (resp., 1) units of 'waste'. The higher number of 'waste' carried by an agent, the higher number of unvisited squares have been entered by that agent. It would not be a very difficult task to design a plan where the agents take turns to act in such a way that all remaining 48 squares are visited in 48 moves. But if the environment gets changed, e.g., is resized or reshaped, the plan must be recalculated. What if we let norms replace plans in this class of environments? Let us investigate the interplay between the agents' utility functions, representing their 'desires', and a normative system which determines their 'room for manouvre'. One idea is to base norms on the spatial relationship between the agents, potentially restricting how the agents may move in the proximity of other agents. We define a condition $Lap_n$, $n \in \{0,1,2,3,4,6,9\}$, with the intended meaning that $Lap_n(x_1,x_2;s)$ holds if and only if the 'protected spheres' of agents $x_1$ and $x_2$ overlap with $i$ squares in a state $s$. The protected sphere con-

Table 1: Transition Type Conditions ($\Lambda$ mnemonic for Leave and $\Omega$ for Oppose).

| $P_k$-operator | Corresponding $C^a_k$-operator | $Prohibited_a(x,s)$ if |
|---|---|---|
| $P_1$ | - | - |
| $P_{2\Lambda}$ | $C^a_{2\Lambda}$ | $d(X_q;s) \wedge \neg d(X_q;a(x,s))$ |
| $P_{2\Omega}$ | $C^a_{2\Omega}$ | $\neg d(X_q;s) \wedge \neg d(X_q;a(x,s))$ |
| $P_{4\Lambda}$ | $C^a_{4\Lambda}$ | $\neg d(X_q;s) \wedge d(X_q;a(x,s))$ |
| $P_{4\Omega}$ | $C^a_{4\Omega}$ | $d(X_q;s) \wedge d(X_q;a(x,s))$ |
| $P_5$ | $C^a_5$ | $\neg d(X_q;a(x,s))$ |
| $P_{6\Lambda}$ | $C^a_{6\Lambda}$ | $\neg(d(X_q;s) \leftrightarrow d(X_q;a(x,s)))$ |
| $P_{6\Omega}$ | $C^a_{6\Omega}$ | $d(X_q;s) \leftrightarrow d(X_q;a(x,s))$ |
| $P_7$ | $C^a_7$ | $d(X_q;a(x,s))$ |

Table 2: Possible changes of $Lap_n$.

| State of affairs | Possible state of affairs in next state |
|---|---|
| $Lap_0(x_1,x_2)$ | $Lap_0(x_1,x_2), Lap_1(x_1,x_2), Lap_2(x_1,x_2), Lap_3(x_1,x_2)$ |
| $Lap_1(x_1,x_2)$ | $Lap_0(x_1,x_2), Lap_1(x_1,x_2), Lap_2(x_1,x_2)$ |
| $Lap_2(x_1,x_2)$ | $Lap_0(x_1,x_2), Lap_1(x_1,x_2), Lap_2(x_1,x_2), Lap_3(x_1,x_2), Lap_4(x_1,x_2)$ |
| $Lap_3(x_1,x_2)$ | $Lap_0(x_1,x_2), Lap_2(x_1,x_2), Lap_3(x_1,x_2), Lap_6(x_1,x_2)$ |
| $Lap_4(x_1,x_2)$ | $Lap_2(x_1,x_2), Lap_4(x_1,x_2), Lap_6(x_1,x_2)$ |
| $Lap_6(x_1,x_2)$ | $Lap_3(x_1,x_2), Lap_4(x_1,x_2), Lap_6(x_1,x_2), Lap_9(x_1,x_2)$ |
| $x_1 \neq x_2 \,\&\, Lap_9(x_1,x_2)$ | $Lap_6(x_1,x_2), Lap_9(x_1,x_2)$ |

sists of ω's square plus the eight surrounding squares. See Fig. 1 in (Hjelmblom, 2008) for an illustration. Table 2 shows how the overlap can change from one state to another, given the nine available actions. Note that since it is always the case that $Lap_9(x_i,x_i)$, $Lap_n(x_1,x_2)$ (Odelstad and Boman, 2004) $x_1 \neq x_2$ for $n < 9$, and $x_1 = x_2$ (Odelstad and Boman, 2004) $Lap_9(x_1,x_2)$. Furthermore, $Lap_n(x_1,x_2)$ (Odelstad and Boman, 2004) $\neg Lap_m(x_1,x_2)$ for $n \neq m$. Now let the 'elementary' conditions $Lap_0$, $Lap_1$, $Lap_2$, $Lap_3$, $Lap_4$, $Lap_6$, together with the 'non-elementary' condition $(\neq \wedge Lap_9)$, form a set of potential descriptive grounds for conditional norms. The set of potential normative consequences corresponding to each ground is constructed by applying the norm-building operators $P_1$, $P_{2\Lambda}$, ..., $P_7$ (see Sect. 1.2) to the conditions listed in the corresponding rows in Table 2. Thus, the potential consequences for, e.g., $Lap_1$ are $P_1Lap_0$,..., $P_7Lap_0$, $P_1Lap_1$,..., $P_7Lap_1$, $P_1Lap_2$,..., $P_7Lap_2$, and $P_1Lap_4$,..., $P_7Lap_4$. Note that it would be meaningless to, e.g., let $P_iLap_4$ be a potential consequence for $Lap_0$, since none of the available acts can change the state of the system in such a way that $Lap_0(x_1,x_2)$ holds in one state and $Lap_4(x_1,x_2)$ holds in the next state.

With these building blocks available, normative systems for Explorer DALMASes can be constructed. Let us use the following approach: For each condition $c$ in the leftmost column of Table 2, one norm $\langle M_1 c, P_i d\rangle$ is added to the normative system for each condition $d$ in the rightmost column.[8] E.g., for $Lap_0$ we add four norms: $\langle Lap_0, P_{k_0} Lap_0\rangle$, $\langle Lap_0, P_{k_1} Lap_1\rangle$, $\langle Lap_0, P_{k_2} Lap_2\rangle$, and $\langle Lap_0, P_{k_3} Lap_3\rangle$. Note that, as regards the ground $(\neq \wedge Lap_9)$, one of $\langle(\neq \wedge Lap_9), P_{k_0} Lap_9\rangle$ and $\langle(\neq \wedge Lap_9), P_{k_1} Lap_6\rangle$ is redundant (since the only conditions that can follow $Lap_9$ are $Lap_9$ and $Lap_6$) and can therefore be removed. This gives a total of 24 norms. Note, however, that not all normative systems formed in this way are *coherent*. To begin with, some sets of rules may be contradictory, according to the intended meaning of the $P_i$ operators, but the problem of coherence (sometimes referred to as 'absence of conflicts') cannot simply be reduced to logical consistency; see for example (Alechina et al., 2013). We will return to this issue in Sect. 3.1.

We would now like to find the 'best' normative system, i.e., the normative system that, together with the simple utility function described earlier, on average makes the Explorer system most efficient. The following measure of 'efficiency' will be employed:

[8] The 'move operator' $M_1$ identifies the agent to which the normative condition applies with the acting agent and with the first agent in the argument sequence $X_v$. See for example (Hjelmblom, 2013) for an explanation. In the following, $M_1$ is omitted to facilitate reading.

the normative system is applied to three different Explorer DALMASes, operating on grids of (almost) equal sizes but different shapes: $6 \times 8$ squares, $7 \times 7$ squares, and $10 \times 5$ squares, respectively. On each grid, three agents are initially placed on square $(1,1)$. A $k$-event run of each of these three systems will be performed, where $k$ is the number of unvisited squares from the beginning. For each run, the ratio between the total number of visited squares and the total number of unvisited squares in the beginning is calculated. If the normative system is not coherent, in the sense that at some point during the run all actions (including stay) become prohibited for the acting agent, then the evaluation score is set to 0. The score of the normative system under evaluation is then the average of the three ratios obtained. We have now obtained an optimisation problem which may be solved with the help of an evolutionary algorithm.

# 3 EVOLUTION OF EXPLORER NORMS

Evolutionary algorithms (EA), being a subfield of evolutionary computation, use the principles of biological evolution (such as reproduction, mutation, recombination, and selection) to solve problems on computers. For a comprehensive introduction to this field the reader is referred to, e.g., (Whitley, 2001). In the Explorer DALMAS setting, there is some randomness in the agents' choices of actions, and in such 'noisy' domains, evolutionary algorithms are known to work well. (Darwen, 2000) We thus implement a basic genetic algorithm (one of the most common forms of EAs) for Explorer DALMAS norms:

1. **Genesis**

   Create an initial population of $n$ candidate normative systems, half of which are entirely randomly generated and half of which consist of $P_1$-consequences (the most permissive consequences) only. Each candidate is represented by a character string consisting of 24 characters from $\{1,...,9\}$.

2. **Evaluation**

   Evaluate each member of the population, by translating the character string to a normative system according to the scheme presented in Sect. 2, running three different systems regulated by this normative system and using as fitness function the average of the evaluation scores of the three runs.

3. **Survival of the Fittest**

   Select a number of members of the evaluated population, favouring those with higher fitness scores, to be the parents of the next generation.

4. **Evolution**

   Generate a new population of offspring by randomly altering and combining elements of the parent candidates. The evolution is performed by the two basic evolutionary operators cross-over and mutation.

5. **Iteration**

   Repeat steps 2-4 until the termination condition (see Table 3) is met.

The evolutionary algorithm was implemented using the Java-based Watchmaker framework for evolutionary computation[9] together with a slightly adapted Java/Prolog implementation of the Waste-collector system (Hjelmblom, 2008; Hjelmblom and Odelstad, 2009).[10] The latter was used in step 2 to perform the $k$-event runs of Explorer systems to be evaluated.

## 3.1 Result

The algorithm was run with the parameter values shown in Table 3; the execution time on an ordinary laptop was 5-6 hours. The graph in Fig. 1 shows the fitness values (evaluation scores) of the best normative system, as well as the average fitness values, in each generation. We can see that, initially, the best fitness (which is obtained by a normative system with $P_1$-consequences only, i.e., a normative system which allows everything) is around 0.78. Up to around generation 25, we can see a slow but quite steady improvement in the best fitness values, although the impact of the slight randomness in the agents' choices of actions is clear. The highest scores, just above 0.86, which roughly corresponds to three more visited squares per run, are obtained in generations 41 and 78. After 25 generations there seems to be no significant improvement.

According to the log, the best normative system in generation 41 (with $P_1$-norms omitted for brevity) is translated to

$$\langle(\neq \wedge Lap_9), P_{2\Omega}Lap_9\rangle, \langle Lap_6, P_{6\Lambda}Lap_9\rangle,$$
$$\langle Lap_4, P_{6\Omega}Lap_4\rangle, \langle Lap_3, P_{2\Lambda}Lap_6\rangle,$$
$$\langle Lap_2, P_{6\Lambda}Lap_4\rangle, \langle Lap_2, P_{4\Omega}Lap_3\rangle,$$
$$\langle Lap_0, P_{4\Omega}Lap_2\rangle, \langle Lap_0, P_{4\Lambda}Lap_1\rangle.$$

[9] http://watchmaker.uncommons.org/

[10] The source code is available for download via http://drp.name/norms/nrtssit, together with a log of a run of the algorithm.


Figure 1: Evolution Progress. Lower curve shows mean fitness.

Table 3: Choice of Parameter Values.

| Parameter | Value |
|---|---|
| Population size | 100 individuals |
| Termination condition | 100 generations evolved |
| Level of elitism | 25% |
| Crossover probability | 0.7 |
| Crossover points | 6 |
| Mutation probability | 0.05 |
| Selection strategy | Roulette wheel selection |

A closer look at the log reveals that, of the best candidates with a fitness over 0.85, (1) all but one (13 out of 14) contain either $\langle Lap_6, P_{6\wedge} Lap_9\rangle$ or $\langle Lap_6, P_{4\wedge} Lap_9\rangle$, and (2) all but three contain $\langle Lap_2, P_{6\wedge} Lap_4\rangle$ or $\langle Lap_2, P_{4\wedge} Lap_4\rangle$. Let us first consider (1). As we see in Table 1, the intended meaning of $\langle Lap_6, P_{6\wedge} Lap_9\rangle$ is that if $Lap_6(x_1,x_2)$ for some agents $x_1$ and $x_2$, then action $a$ is prohibited for the moving agent if $\neg Lap_9(x_1,x_2;s) \wedge Lap_9(x_1,x_2;a(x,s))$ or $Lap_9(x_1,x_2;s) \wedge \neg Lap_9(x_1,x_2;a(x,s))$. Since $Lap_6$ implies $Lap'_9$, the second disjunct never becomes true when $Lap_6(x_1,x_2)$; hence $a$ is prohibited for the moving agent if $Lap_6(x_1,x_2;s)$ and $Lap_9(x_1,x_2;a(x,s))$. The meaning of $\langle Lap_6, P_{4\wedge} Lap_9\rangle$ is that if $Lap_6(x_1,x_2)$ for some agents $x_1$ and $x_2$, then $a$ is prohibited if $\neg Lap_9(x_1,x_2;s) \wedge Lap_9(x_1,x_2;a(x,s))$; i.e., if $Lap_6(x_1,x_2;s)$ and $Lap_9(x_1,x_2;a(x,s))$. Hence, $\langle Lap_6, P_{6\wedge} Lap_9\rangle$ and $\langle Lap_6, P_{4\wedge} Lap_9\rangle$ are 'operationally equivalent' in the Explorer DALMAS setting, in the sense that they prohibit the same actions in the same situation. Furthermore, both are operationally equivalent to $\langle Lap_6, P_7 Lap_9\rangle$ with the intended interpretation that if $Lap_6$ then the moving agent shall see to it that not $Lap_9$. A similar case can be made for (2); $\langle Lap_2, P_{6\wedge} Lap_4\rangle$, $\langle Lap_2, P_{4\wedge} Lap_4\rangle$ and $\langle Lap_2, P_7 Lap_4\rangle$ are operationally equivalent and thus interchangeable in this setting.

(1) and (2) illustrate that, in many settings, the set of consequences may contain redundancy. This is an effect of the fact that, in this particular setting, the set of grounds and the set of consequences are constructed from the same set of conditions. Whether this is a problem or not is probably dependent on the particular setting. We may also note that, for example, the meaning of $\langle Lap_0, P_{4\Omega} Lap_2\rangle$ would be that if $Lap_0(x_1,x_2)$ for some agents $x_1$ and $x_2$, then $a$ is prohibited for the moving agent if $Lap_2(x_1,x_2;s) \wedge Lap_2(x_1,x_2;a(x,s))$. Now, since $Lap_0$ implies $Lap'_2$, $Lap_2(x_1,x_2;s) \wedge Lap_2(x_1,x_2;a(x,s))$ can never become true when $Lap_0(x_1,x_2)$. Hence, $\langle Lap_0, P_{4\Omega} Lap_2\rangle$ will never prohibit any actions, and is thus operationally equivalent to, $\langle Lap_0, P_1 Lap_2\rangle$ in this setting. This illustrates another kind of redundancy. Another consequence of employing negative permission is that normative systems may evolve which are incoherent (see Sect. 1.2) according to the underlying logic of the $P_k$ operators, but still meaningful in an 'operational' sense. To avoid or at least reduce redundancy and logical incoherence (and thus, potentially, significantly reduce the search space for the evolutionary algorithm) in the setting at hand, a more precise representation of genes and a more careful design (based on a more thorough analysis of the relationships between potential grounds and consequences) of the genetic opera-

tors is required. For this purpose, the mechanisms for norm addition and subtraction described in (Lindahl and Odelstad, 2013, Sect. 4.3) might be very useful.

Based on the above analysis, the following set of Explorer norms (again, $P_1$-norms are omitted) is suggested: $\{\langle M_1Lap_6, P_7Lap_9\rangle, \langle M_1Lap_2, P_7Lap_4\rangle\}$. The intended interpretation is

(1) For all $x,y$: $M_1Lap_6(x,y;x,s) \rightarrow P_7Lap_9(x,y;x,s))$; and

(2) For all $x,y$: $M_1Lap_2(x,y;x,s) \rightarrow P_7Lap_9(x,y;x,s))$.

This represents the following simple set of 'rules of thumb': (1) If you stand in the square next to another agent's square, you shall move so that you do not end up in the same location as the other agent, and (2) if your protected sphere overlaps another agent's protected sphere with two squares, you shall move so that the overlap does not increase to four. These rules may be expressed in logical form using the deontic operator *Shall* and the action operator *Do*: (1) $\forall x,y$: $Lap_6(x,y)$, and $x$ is the moving agent, implies $Shall\,Do(x,\neg Lap_9(x,y))$; and (2) $\forall x,y$: $Lap_2(x,y)$, and $x$ is the moving agent, implies $Shall\,Do(x,\neg Lap_4(x,y))$. Cf. (Odelstad and Boman, 2004; Hjelmblom, 2014b).

Test runs indicate that the average improvement with this very simple normative system compared with a system with no restrictions is two to three additional squares visited. As the Explorer DALMAS example was chosen for demonstration purposes only, we shall be content with the simple analysis performed here. In more complex scenarios, other more powerful (e.g., statistical) methods could be useful.

## 3.2 Discussion

Validation of the suggested approach to the design of normative systems for problem-solving MAS is, of course, a non-trivial problem. One aspect of this problem is the difficulty of applying this approach, but most important is probably to focus on the quality of the results it produces, i.e., to validate the systems obtained by applying the approach. The performance of norm-regulated MAS designed in this way could, for example, be compared with the performance of systems (norm-regulated systems as well as, e.g., planning systems) designed 'by hand'. Such comparisons require domain-specific performance measures, which makes a general-level (i.e., domain independent) validation very difficult, if not impossible. Even within a specific domain, validation is non-trivial and sensitivity analyses are required. A good starting-point is to consider every tool in the evolutionary toolbox, together with a thorough analysis of the domain at hand, to increase the chance of evolving the optimal normative system. First, the parameters controlling the evolutionary algorithm may be varied: the population size, the number of evolved generations, the level of elitism (i.e., the portion of the best candidates which are allowed to survive into the next generation), the probability of crossover, the number of crossover points, and the selection strategy (e.g., tournament selection instead of roulette wheel). Other ideas include using other representations of chromosomes, such as tree-based representations to allow for normative systems with a variable number of norms, or (as has already been mentioned) more carefully designed evolutionary operators that exclude redundant and/or incoherent candidates from evaluation. More advanced schemes, such as island evolution (where several populations are evolved in parallel, with a small probability of 'migration' between such 'islands') or cooling (where the crossover and mutation probabilities gradually decreases), could also be tried.

Furthermore, the parameters for the particular setting may also be varied. For example, one might want to consider grounds and consequences based on other conditions. In the Explorer DALMAS domain one could try, e.g., $Lap_n$ conditions based on larger protected spheres (since it seems reasonable to expect that a normative system based on small protected spheres will be most 'effective' when the agents are relatively close to each other), or generalised versions of $Lap_n$ conditions involving three or more agents. Other ideas are to allow individual utility functions for each agent, or evolving the utility function and the normative system in parallel. In general, special treatment is required for domains such as the Explorer DALMAS where the fitness evaluations are 'noisy', i.e., subject to some degree of randomness. To deal with noisy fitness evaluations, a number of techniques are available, for example increasing the population size, and resampling and averaging the fitness. (Di Pietro et al., 2002, Sect. 3.3) As described in Sect. 2, a variant of the latter technique is used in the Explorer DALMAS fitness evaluations. Another option regarding the evaluation function is to allow more or less variation regarding, e.g., grid sizes or shapes, number of agents, number of events per run and number of runs per normative system. However, large populations, in combination with expensive fitness calculations in each generation, are computationally challenging. The *moving average* approach by Di Pietro et al. can be used to reduce the number of samples needed per generation, and thus allow for running more generations in a given run-time. When a candidate is generated for the first time, its 'fitness array' is initialised with $n$ fitness evaluations. For

each new generation, the evaluation score is calculated only once, and the oldest score in the fitness array is replaced with the new score. A candidate's fitness is then the average of the evaluation scores in the fitness array.

# 4 CONCLUSION AND FUTURE WORK

A sketch of a methodology for using evolutionary mechanisms as part of the pre-runtime design of normative systems for problem-solving MAS was presented. The idea behind this methodology is to use a 'top-down' approach of selecting (a subset of ) the most 'efficient' norms from an evolved normative system, rather than a 'bottom-up' approach of designing a normative system entirely from scratch. To illustrate the idea, a simple system, based on the DALMAS architecture for norm-regulated MAS was employed as part of the evaluation step of an evolutionary algorithm. The results show that an evolutionary algorithm has the potential of being a useful tool when designing normative systems for problem-solving MAS.

Ideas for future work include trying to formalise and further investigate the notion of operational equivalence which was introduced in Sect. 3.1. Also left for future work is further validation of the suggested methodology, for example by applying the methodology in other domains in which the grounds of the norms and the consequences are based on different sets of descriptive conditions, or by further validating the evolved normative system for the Explorer DALMAS. One could experiment with different domain-specific parameters as well as evolutionary algorithm parameters, as suggested in Sect. 3.2, to see if better solutions can be found and thus gain more support for the ideas suggested here. It could be interesting to, e.g., explore variable-sized normative systems and evaluation functions which impose a 'penalty' for large normative systems, since in many cases it could be desirable to rely on a small number of 'rules of thumb' and avoid overly complex normative systems which may become expensive in terms of calculations. Investigating the possibility to design more 'accurate' evolutionary operators also seems like a promising idea.

# ACKNOWLEDGEMENTS

The author is very grateful to Jan Odelstad and Magnus Boman for valuable ideas and suggestions.

# REFERENCES

Alechina, N., Bassiliades, N., Dastani, M., Vos, M. D., Logan, B., Mera, S., Morris-Martin, A., and Schapachnik, F. (2013). Computational Models for Normative Multi-Agent Systems. In Andrighetto, G., Governatori, G., Noriega, P., and van der Torre, L. W. N., editors, *Normative Multi-Agent Systems*, volume 4 of *Dagstuhl Follow-Ups*, pages 71–92. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany.

Andrighetto, G., Castelfranchi, C., Mayor, E., McBreen, J., Lopez-Sanchez, M., and Parsons, S. (2013a). (Social) Norm Dynamics. In Andrighetto, G., Governatori, G., Noriega, P., and van der Torre, L. W. N., editors, *Normative Multi-Agent Systems*, volume 4 of *Dagstuhl Follow-Ups*, pages 135–170. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany.

Andrighetto, G., Governatori, G., Noriega, P., and van der Torre, L. W. (2013b). Normative multi-agent systems, volume 4 of dagstuhl follow-ups. *Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik*.

Balke, T., Cranefield, S., Tosto, G. D., Mahmoud, S., Paolucci, M., Savarimuthu, B. T. R., and Verhagen, H. (2013). Simulation and NorMAS. In Andrighetto, G., Governatori, G., Noriega, P., and van der Torre, L. W. N., editors, *Normative Multi-Agent Systems*, volume 4 of *Dagstuhl Follow-Ups*, pages 171–189. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany.

Darwen, P. (2000). Computationally intensive and noisy tasks: co-evolutionary learning and temporal difference learning on backgammon. In *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, volume 2, pages 872–879 vol.2.

Di Pietro, A., While, R. L., and Barone, L. (2002). Learning in robocup keepaway using evolutionary algorithms. In *GECCO*, volume 2, pages 1065–1072.

Hjelmblom, M. (2008). Deontic action-logic multi-agent systems in Prolog. Technical Report 30, University of Gävle, Division of Computer Science.

Hjelmblom, M. (2011). State transitions and normative positions within normative systems. Technical Report 37, University of Gävle, Department of Industrial Development, IT and Land Management.

Hjelmblom, M. (2013). Norm-regulated transition system situations. In Filipe, J. and Fred, A., editors, *Proceedings of the 5th International Conference on Agents and Artificial Intelligence*, ICAART 2013, pages 109–117, Portugal. SciTePress.

Hjelmblom, M. (2014a). Instrumentalization of norm-regulated transition system situations. In Filipe, J. and Fred, A., editors, *Agents and Artificial Intelligence*, volume 449 of *Communications in Computer and Information Science*, pages 80–94. Springer Berlin Heidelberg.

Hjelmblom, M. (2014b). Normative positions within norm-regulated transition system situations. In *Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2014 IEEE/WIC/ACM International Joint Conferences on*, volume 3, pages 238–245.

Hjelmblom, M. and Odelstad, J. (2009). jDALMAS: A Java/Prolog framework for deontic action-logic multi-agent systems. In Håkansson, A., Nguyen, N., Hartung, R., Howlett, R., and Jain, L., editors, *Agent and Multi-Agent Systems: Technologies and Applications*, volume 5559 of *Lecture Notes in Computer Science*, pages 110–119. Springer Berlin / Heidelberg. doi:10.1007/978-3-642-01665-3_12.

Lindahl, L. (1977). *Position and change: a study in law and logic*. Synthese library. D. Reidel Pub. Co.

Lindahl, L. and Odelstad, J. (2004). Normative positions within an algebraic approach to normative systems. *Journal of Applied Logic*, 2(1):63 – 91.

Lindahl, L. and Odelstad, J. (2013). The theory of joining-systems. In Gabbay, D., Horthy, J., Parent, X., van der Meyden, R., and van der Torre, L., editors, *Handbook of Deontic Logic*, volume 1, chapter 9, pages 545–634. College Publications, London.

Luke, S., Hohn, C., Farris, J., Jackson, G., and Hendler, J. (1998). Co-evolving soccer softbot team coordination with genetic programming. In Kitano, H., editor, *RoboCup-97: Robot Soccer World Cup I*, volume 1395 of *Lecture Notes in Computer Science*, pages 398–411. Springer Berlin Heidelberg.

Nakashima, T., Takatani, M., Udo, M., and Ishibuchi, H. (2004). An evolutionary approach for strategy learning in robocup soccer. In *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, volume 2, pages 2023–2028. IEEE.

Odelstad, J. (2008). *Many-Sorted Implicative Conceptual Systems*. PhD thesis, Royal Institute of Technology, Sweden. QC 20100901.

Odelstad, J. and Boman, M. (2004). Algebras for agent norm-regulation. *Annals of Mathematics and Artificial Intelligence*, 42:141–166. doi:10.1023/B:AMAI.0000034525.49481.4a.

Verhagen, H. and Boman, M. (1999). Norms can replace plans. In *IJCAI'99 Workshop on Adjustable, Autonomous Systems*.

Whitley, D. (2001). An overview of evolutionary algorithms: practical issues and common pitfalls. *Information and Software Technology*, 43(14):817 – 831.