Understanding the Emergence of Conventions in Multi-Agent Systems

Adam Walker and **Michael Wooldridge***

Department of Computing
Manchester Metropolitan University
Chester Street, Manchester M1 5GD
United Kingdom

Abstract

In this paper, we investigate techniques via which a group of autonomous agents can reach a global agreement on the use of social conventions by using only locally available information. Such conventions play a central role in naturally-occurring social systems, and there are good reasons for supposing that they will play a similarly important role in artificial social systems. Following a short review of conventions and their use in distributed artificial intelligence, we present a formal model that rigorously defines both our experimental methodology, and the performance measures we use to quantify the success of our experiments. We then describe sixteen different mechanisms for bringing about agreement on conventions, and present experimental results obtained for each of these methods. A tentative analysis of these results is given, and the paper concludes with some comments and issues for future work.

Topic areas: organization self-design, cooperation.

*main contact — email: M.Wooldridge@doc.mmu.ac.uk

1 Introduction

Recent work in Distributed Artificial Intelligence (DAI) has investigated the possibility of using norms, conventions, and social laws in multi-agent systems. Examples of the issues investigated include the control of aggression [1], how conventions might emerge within agent societies [7], the role of social structure in the emergence of conventions [5], group behaviour [2], and the reconsideration of commitments [4]. In addition, researchers working in philosophy, sociology, and economics have considered similar issues. A good example is the work of Lewis [6], who has made some progress towards a (non-formal) theory of normative behaviour.

Conventions play a key role in the social process. They provide agents with a template upon which to structure their action repertoire. They represent a behavioural constraint, striking a balance between individual freedom on the one hand, and the goal of the agent society on the other. As such, they also simplify an agent's decision making process, by dictating courses of action to be followed in certain situations. One key issue in the understanding of conventions is to decide on the most effective method by which they can come to exist within an agent society. There are two main approaches:

Off-line design: In this approach, social laws are designed off-line, and hard-wired into agents. Examples in the DAI literature include [8, 3, 1].

Emergence from within the system: This possibility is investigated by [7, 5], who experiment with a number of techniques by which a convention can 'emerge' from within a group of agents.

The first approach will often be simpler to implement, and might present the system designer with a greater degree of control over system functionality. However, there are a number of disadvantages with this approach. First, it is not always the case that *all* the characteristics of a system are known at design time; this is most obviously true of *open systems.* In such systems, the ability of agents to organise themselves would be advantageous. Secondly, in complex systems, the goals of agents (or groups of agents) might be constantly changing. To keep reprogramming agents in such circumstances would be costly and inefficient. Finally, the more complex a system becomes, the less likely it is that system designers will be able to design effective social laws. Here, flexibility from within the agent society might result in greater coherence.

This paper is thus concerned with the latter approach — that of emergence from within the society. Specifically, we investigate the efficiency of various mechanisms via which a group of autonomous agents can come to reach a *global* agreement on the use of social conventions by using only *locally available* information. Thus each agent must decide on which convention to adopt based solely on its own experiences, as recorded in its internal state; pre-defined inter-agent power structures or authority relationships are not allowed. The key problem is to design a *strategy update function*, representing an agent's decision-making process, that, when used by every agent in the society, will bring the society to a global agreement as efficiently as possible. We begin, in the following section, by presenting a formal model that rigorously defines the problem domain, our methodology, and the performance measures we use to assess the effectiveness of strategy update functions. In section 3, we describe our experimental environment, and present sixteen different strategy update functions, whose efficiency we have evaluated by experiment. Our results are presented in section 4, and some conclusions appear in section 5. Note that this work builds on that of [7, 1, 5].

2 The Formal Model

We begin by assuming a set $Ag = \{1, \ldots, l\}$ of *agents*. At any time, an agent is assumed to have chosen one of a set $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$ of *strategies*. These strategies represent the possible norms or conventions that an agent may fix upon. Exactly what these strategies *are* is not significant, at least for the purposes of this formal model. Initially, each agent is assigned an element of Σ at random.

An *interaction* occurs when two agents compare strategies. Formally, we define the set I, of all interactions, by $I = \Sigma \times \Sigma$. We always consider an interaction $\iota \in I$ from the point of view of some particular agent; we write $self(\iota)$ for the strategy chosen by this agent in the interaction ι, and $other(\iota)$ for the strategy chosen by the other participant in ι.

An agent's *memory* consists of a finite sequence of interactions, representing those that the agent has been involved in. Formally, the set M of all memory states is defined by $M = I^*$. If $m \in M$, then $|m|$ denotes the length of m; if $n \in \{0, \ldots, |m| - 1\}$, then $m(n)$ denotes the n'th interaction in m. If $\sigma \in \Sigma$ and $m \in M$, then $obs(\sigma, m)$ is the number of times that other agents have been seen to use σ in m:

$$obs(\sigma, m) \quad = \quad |\{n \mid n \in \{0, \ldots, |m| - 1\} \text{ and } other(m(n)) = \sigma\}|$$

Note that in some domains, (such as the one we describe in section 3), there may be times at which an agent is not involved in an interaction, but is performing some other kind of action. However, these non-interactive 'moves' play no part in the strategy selection process, and may therefore be ignored in the formal model.

Our aim in this paper is to investigate the properties of *strategy update functions*. A strategy update function u has the signature $u : M \rightarrow \Sigma$, i.e., it takes a memory state, and, on the basis of the experiences recorded in this state, selects a strategy. Let U be the set of all strategy update functions.

A *run*, r, is a function $r : Ag \times \mathbb{N} \rightarrow I$, that must satisfy the following invariant property: $\forall n \in \mathbb{N}$, there exists an irreflexive bijection $f_n : Ag \rightarrow Ag$, such that $\forall i \in Ag$, if $self(r(i,n)) = \sigma$ and $other(r(i,n)) = \sigma'$, then $self(r(f_n(i),n)) = \sigma'$ and $other(r(f_n(i),n)) = \sigma$. A run r represents one possible *history* of a system; for each agent $i \in Ag$, and for each time point $n \in \mathbb{N}$, it assigns i an interaction $r(i,n)$ representing that which i was involved in at time n. The invariant on runs ensures that interactions are *symmetrical*, i.e., that two agents are involved in every interaction. Let R be the set of all runs.

We let $mem(i,r,n)$ denote the memory of agent $i \in Ag$ in run $r \in R$ up to time $n \in \mathbb{N}$. We can visualise $mem(i,r,n)$ as the sequence $(r(i,0), r(i,1), \ldots, r(i,n))$. Note that, for simplicity, we have here assumed that agents have unbounded memory. We say that a run $r \in R$ is consistent with the use of strategy update function $u \in U$, (notation $uses(r,u)$), iff $\forall i \in Ag$, $\forall n \in \mathbb{N}$, if $r(i,n+1) = \iota$, then $self(\iota) = u(mem(i,r,n))$. We denote by R_u that subset of R whose members are consistent with the use of u, i.e., $R_u = \{r \mid r \in R \text{ and } uses(r,u)\}$. The set R_u represents the characteristic behaviour of the strategy update function u. With respect to R_u, we can define a number of *performance measures*, via which the effectiveness of u can be measured.

Our aim when designing a strategy update function $u \in U$ is that, by using it, the agents in a system will come to *converge* on a strategy. We begin by formalising the notion of convergence. We denote by $chosen(\sigma,n,r)$ the set of agents that have chosen strategy $\sigma \in \Sigma$ at time $n \in \mathbb{N}$ in run $r \in R$, i.e., $chosen(\sigma,n,r) = \{i \mid i \in Ag \text{ and } self(r(i,n)) = \sigma\}$. We then define the *convergence* of a run $r \in R$ at time $n \in \mathbb{N}$, (denoted $conv(r,n)$), to be the fraction of agents using the most popular strategy at time n in run r.

$$conv(r,n) = \frac{\max\{|chosen(\sigma,r,n)| \mid \sigma \in \Sigma\}}{|Ag|}$$

If $|\Sigma| = 2$, then convergence will range from 0.5 to 1.0; if $|\Sigma| = 3$, then convergence will range from 0.33 to 1.0. If $conv(r,n) = 1.0$, then in run r at time n, every agent has chosen the same strategy.

If $u \in U$, then we let $\mathcal{C}^u_n$ denote the average convergence of all runs $r \in R_u$ at time $n \in \mathbb{N}$[1]. Note that we normally express $\mathcal{C}^u_n$ as a percentage. The value $\mathcal{C}^u_n$, for some $n \in \mathbb{N}$, is the first measure by which we assess the performance of a strategy update function; we aim to design a function $u \in U$ that will make $\mathcal{C}^u_n = 100$ for as small n as possible.

The time it takes for a strategy update function to bring about convergence is not the only performance measure that we can use. Another important criterion is the *average number of strategy changes per interaction*. The intuition behind this measure is that, in real-world situations, changing from one strategy to another generally incurs a cost. Consider, for example, the financial and human resources required for an organisation to move from one computer operating system to another. Let $sc(i,r,n)$ be the number of times agent $i \in Ag$ changes strategy in run $r \in R$ up to time $n \in \mathbb{N}$:

$$sc(i,r,n) = |\{o \mid o \in \mathbb{N}, o < n, \text{ and } self(r(i,o)) \neq self(r(i,o+1))\}|$$

The average number of strategy changes per interaction in run $r \in R$ up to time $n \in \mathbb{N}$ is then given by $changes(r,n)$:

$$changes(r,n) = \frac{\sum_{i \in Ag} sc(i,r,n)}{n.|Ag|}$$

We then let $\mathcal{H}^u_n$ denote the average number of strategy changes per interaction up to time $n \in \mathbb{N}$ over each run $r \in R_u$, for strategy update function $u \in U$. The value $\mathcal{H}^u_n$ will range from 0.0 to 1.0; if $\mathcal{H}^u_n = 1.0$, then every agent changes strategy after every interaction. Lower values of $\mathcal{H}^u_n$ are thus preferable. The value $\mathcal{H}^u_n$ will be our second measure of strategy update function performance.

[1] This notation was introduced by [5].

Of particular interest to us are strategy update functions that guarantee *stable convergence*. A function u has this property iff $\forall r \in R_u$, $\exists n \in \mathbb{N}$ such that $\forall o, p \in \mathbb{N}$, and $\forall i, j \in Ag$, if $o \geq n$ and $p \geq n$ then $self(r(i, o)) = self(r(j, p))$; in other words, if u guarantees that the agents will come to agree on a strategy, and then remain fixed on it. Note that if u guarantees stable convergence, then $\mathcal{H}_n^u \rightarrow 0.0$ as $n \rightarrow \infty$.

As a third, and final performance measure, we consider the *maximum* number of strategy changes that any agent incurs on any run $r \in R_u$ up to time n. We denote this value by $\mathcal{M}_n^u$. Obviously, low values of $\mathcal{M}_n^u$ are preferable.

This completes our formal model. We can now describe our experimental methodology. Our aim is to compare the effectiveness of a number of strategy update functions, using the measures $\mathcal{C}_n^u$, $\mathcal{H}_n^u$, and $\mathcal{M}_n^u$. To do this, we take each strategy update function u, and generate, by computer simulation, a set of runs of u, that we hope are representative of R_u. We then analyse these runs, to empirically determine the values $\mathcal{C}_n^u$, $\mathcal{H}_n^u$, and $\mathcal{M}_n^u$.

3 The Experimental Environment

In this section, we describe both our experimental environment and the sixteen different strategy update functions that we investigate. Our experimental environment is based on that of [1]. In this environment, agents move about a grid in search of food. They are endowed with a budget that is increased by eating an item of food, but decreased by movement around the grid. Agents therefore have the goal of eating as much food as possible in order to maximise their budget. Agents can move only one square at a time in either a horizontal or vertical direction. Likewise, at any one time, they can 'see' only one square in each of these directions. If they see an item of food in a square, then they move to that square and eat the food. Otherwise, they move to a randomly selected empty square in search of food. Naturally, more than one agent may at any time make a bid for the same item of food. If this happens, then one of the agents is selected randomly. However, agents can also attack others who are in possession of a food item. This aggression costs both aggressor and victim a large slice of their budget, but may be rewarding if the aggressor wins the food item. The winner in any confrontation is always the agent with the highest budget.

The purpose of the experiments in [1] was to examine how aggression might be reduced by the introduction of conventions into the system. Two conventions

were identified — one of ownership, where agents cannot attack those eating their own food, and another where agents cannot attack others who are on their right-hand side. Although these experiments provide a valuable insight into normative behaviour, the conventions were nevertheless hard-wired into agent's architectures. Our experiments are concerned with how such conventions might come to exist within the agent society using observation and a strategy update function. We want to provide our agent society with a variety of possible strategies by which to reduce aggression, and a suitable update function, and measure how effectively the agents come to agree on a single strategy or convention.

To achieve this, we have chosen the latter strategy of [1] — that of giving precedence to agents in a particular location — but have extended it to consist of four possible strategies: precedence to those on the right, to those above, to those below, and to those on the left. Agents begin an experiment with one of these strategies chosen at random, and as the experiment progresses, they observe the strategies adopted by other agents. Using an update function, they may modify their chosen strategy as they interact with other agents.

Some other minor modifications were needed to the world of [1], which, whilst maintaining the functionality of the original system, extended it to allow more meaningful results to be obtained within the context of our research goals. First, 100 agents were used, rather than the 12 of [1]. The agent to food ratio stays the same at 2:1, giving a total of 50 food items at the start of each experiment. We also use a larger world. The grid is a square of 225 cells — 15×15. Finally, we use longer experiments. In [1], a *match* is over when all the food has been eaten, or all the agents have 'died'. (An agent dies when its budget falls below one.) Matches do not allow sufficient time to assess the processes of norm evolution, and so we extended each *experiment* to consist of 50 matches. At the end of each match the food supply is renewed at random locations, and all dead agents are replaced by new agents at the same location. Note that new agents do not retain the memories of their predecessors, and their normative strategy is assigned afresh.

3.1 Basic Strategy Update Functions

We have evaluated four different basic strategy update functions in order to measure their effectiveness in producing convergence, using the experimental framework described above. These basic strategy update functions are as follows[2]:

[2]The first three are adapted from [7]; the fourth is entirely new.

Simple majority: This is the simplest form of update function. Agents will change to an alternative strategy if so far they have observed more instances of it in other agents than their present strategy. If more than one strategy has been observed more than that currently adopted, the agent will choose the strategy observed most often.

Using the notation introduced in section 2, we may define this strategy update function as follows:

$$u(m) = \sigma \text{ iff } \nexists \sigma' \in \Sigma \text{ such that } obs(\sigma', m) > obs(\sigma, m).$$

Simple majority with memory restart: This function is essentially the same as simple majority, except that when an agent has changed strategy, he clears his memory. (Note that the signature of this strategy update function is thus $u : M \rightarrow \Sigma \times M$.)

Simple majority with communication by agent type: Agents are divided into two types. As well as observing each other's strategies, agents in these experiments can communicate with others whom they can 'see', and who are of the same type. When they communicate, they exchange memories, and each agent treats the other agent's memory as if it were his own, thus being able to take advantage of another agent's experiences. By restricting communication to agents of the same type, we apply the 'extroversion radius' of [7]. In other words, agents are particular about whom they confide in.

Simple majority with communication on success: Here we use a utility function that employs a form of communication based on a success threshold. When an individual agent has reached a certain level of success with a particular strategy, he communicates his memory of experiences with this successful strategy to all other agents that he can 'see'. Note, only the memory relating to the successful strategy is broadcast, not the whole memory. The intuition behind this update function is that an agent will only communicate with another agent when it has something *meaningful* to say. This prevents 'noise' communication. In our experiments, the threshold chosen was a strategy where the total observed was greater than 20, but was also at least twice as much as any other strategy total.

3.2 External Majority Variations

The above strategy update functions, whilst differing in certain key aspects, all cause agents to change strategy on the basis of a simple majority. This is an intuitively appealing approach, and certainly performed significantly better than the other update functions tested by Shoham and Tennenholtz [7]. However, we had certain reservations about how accurately this approach accorded with our own understanding of normative behaviour. Our concerns were:

- This methodology seems to be more akin to the phenomenon of imitation, rather than norm convergence. Imitation represents a subconscious copying of another's actions, rather than a reasoned choice. Imitation can lead to negative results — as in the case of riots. Thus, this function may not guarantee that the best strategy is adopted. It could be argued that in our experiments, no one strategy is preferable; the important point is that all agents adopt the same one. However, in many instances — e.g., in the world of standards — certain strategies may be preferable to others.

- The simple majority may lead to agents changing strategy frequently. As we pointed out in section 2, frequent strategy changes are, in general, inefficient. A better approach might be to change strategy only if it is observed in considerably more agents than the current one.

Given these concerns, we decided to experiment with variations of the simple majority update function, using majorities of greater than one. The intention was to observe to what degree higher majorities affected the value of $\mathcal{C}_n^u$, but to balance this against any improvements in the values of $\mathcal{H}_n^u$ and $\mathcal{M}_n^u$. In addition to the simple majority update function, the following three variations were tested:

Double majority: Agents changed from strategy σ to strategy σ' if the number of observations for σ' was more than twice that for σ.

Quadruple majority: Agents changed from strategy σ to strategy σ' if the number of observations for σ' was more than four times that for σ.

Dynamic majority: In this variation, the size of the majority is a function of each agent's total number of strategy changes to date. If an agent has changed strategy only once, then he will do so again should he observe a simple majority of one against an alternative strategy. If he has changed strategy

twice, then he will not change again until he observes a total of more than double for an alternative strategy — and so on. Using this variation, agents become more reluctant to change as their number of strategy changes increases. Note that with this variation, it is technically possible for a deadlock situation to arise, in which the system becomes fixed in a steady state. However, we have not observed such a situation in practice.

4 Experimental Results

The four basic update functions described in section 3.1 were tested, using the four variations described in section 3.2. This resulted in sixteen different strategy update functions in total. Each of these functions was run for 50 matches, with a match lasting, on average, for six interactions. Thus, the length n of each run generated in each experiment was approximately 300. To avoid statistical anomalies, each experiment was repeated 80 times. The remaining parameters for the experiments were as follows: agents can choose from 4 strategies, and so $|\Sigma| = 4$. Each experiment consisted of 100 agents, and so $|Ag| = 100$.

4.1 Norm Convergence — $\mathcal{C}_n^u$

The experimentally determined values of $\mathcal{C}_n^u$, (the average convergence at time n), for $n \simeq 300$, are given in Table 1. Note that these values are expressed as percentages. Rows (1–4) correspond to the four basic update functions described in section 3.1, and columns (a–d) correspond to variants in the size of majority, described in section 3.2.

We make the following observations on these results:

- By increasing the size of the majority, we appear to see a slight degradation in $\mathcal{C}_n^u$. However, the significance of this is not clear. Certainly, there is very little difference between a majority size of more than double, and one of more than four times as much.

- In [7], the two update functions that performed best with respect to $\mathcal{C}_n^u$ were the memory restart function (2a), and the communication by agent type function (3a). Both of these used a simple majority, (i.e., a majority of at least one), and performed significantly better than using the simple majority with full memory function (1a). We had similar expectations for our

Update function	Size of majority			
	(a) Simple	(b) Double	(c) Quadruple	(d) Dynamic
(1) External majority	75	73	74	84
(2) Memory restart	82	77	76	77
(3) Communication by type	76	75	74	79
(4) Communication on success	99	80	73	86

Table 1: Experimentally Determined Values of $\mathcal{C}_n^u$, for $n \simeq 300$

experiments, but these do not appear to have been fully borne out. Whilst (2a) has shown some improvement, (3a) shows no significant gain over a function using no communication at all.

- The dynamic majority variation appears overall to be the most efficient of variations (b–d) with respect to $\mathcal{C}_n^u$. In two experiment sets — (1d) and (3d) — it performs even better than the simple majority function.

- Our own update function, (4), produces the best results for $\mathcal{C}_n^u$. All experiments using the simple majority reached a value for $\mathcal{C}_n^u$ of either 100% or 99% for $n \simeq 300$. We are unsure at present as to the processes that lead to such a good result for norm convergence. However, (4) also produces the most significant differences amongst the three majority variations of any of the functions. Here, increasing the majority that agents use does have a significant effect on $\mathcal{C}_n^u$.

4.2 Strategy Change Results — $\mathcal{H}_n^u$ and $\mathcal{M}_n^u$

One possible drawback to the simple majority function is the number of strategy changes that agents might make; as we noted in section 2, changes in strategy can be costly, and lead to inefficiencies within a system. For this reason, we introduced $\mathcal{H}_n^u$, the average number of strategy changes per interaction, which we use as the second measure of strategy update function efficiency. Table 2 shows, for each of the sixteen strategy update functions investigated, the experimentally determined values of $\mathcal{H}_n^u$, for $n \simeq 300$. Table 3 shows, for each of the sixteen strategy update

Update function	Size of majority			
	(a) Simple	(b) Double	(c) Quadruple	(d) Dynamic
(1) External majority	0.021	0.016	0.012	0.019
(2) Memory restart	0.027	0.021	0.019	0.026
(3) Communication by type	0.062	0.026	0.014	0.032
(4) Communication on success	0.022	0.012	0.010	0.022

Table 2: Experimentally Determined Values of $\mathcal{H}_n^u$, for $n \simeq 300$

Update function	Size of majority			
	(a) Simple	(b) Double	(c) Quadruple	(d) Dynamic
(1) External majority	9	5	3	5
(2) Memory restart	20	11	10	11
(3) Communication by type	57	24	6	7
(4) Communication on success	26	4	3	4

Table 3: Experimentally Determined Values of $\mathcal{M}_n^u$, for $n \simeq 300$

functions, the experimentally determined values of $\mathcal{M}_n^u$, (the maximum number of strategy changes made by any agent up to time n), again for $n \simeq 300$.

Some observations we can make from these results are as follows:

- Using the simple majority function, (1a), leads to a high value of $\mathcal{H}_n^u$, as expected. However, (1a) is by no means the worst. The two modifications to (1a) that were described in [7] — experiments (2a) and (3a) — also perform poorly and in particular, communication by agent type (3a) leads to a very high value of $\mathcal{H}_n^u$. The update function (4), which performed so well with respect to $\mathcal{C}_n^u$, also performs poorly with respect to $\mathcal{H}_n^u$.

- Increasing majority size does lead to an improvement in $\mathcal{H}_n^u$, as expected. However, in running the experiment using a dynamically determined majority, we expected an even better performance. This has not proved to be

the case. The performance of dynamically determined majority appears to be comparable to that of the simple majority variation. The value of $\mathcal{M}_n^u$ for the dynamically determined majority is, however, more encouraging. It is either equal to or better than that recorded against the double majority.

- The most significant differences across the four columns of Tables 2 and 3 occur in the two experiment sets that deal with communication.

Clearly, those update functions that perform best with respect to $\mathcal{C}_n^u$ do not necessarily perform as well with respect to $\mathcal{H}_n^u$ and $\mathcal{M}_n^u$. What is needed is a trade-off between the speed by which agents can come to agree on a norm, and the number of changes of strategy they incur in the process. Combining all three tables, perhaps the best experiment set overall is (4b).

5 Concluding Remarks

Our research has been motivated by a concern, voiced in [7], that an understanding of the mechanisms of convention evolution will be important for the design of future multi-agent systems. Building on the work of [7, 1, 5], we have presented a new formal model that defines the problem domain, a methodology, and a set of performance measures. We have described sixteen mechanisms by which agents can come to a global agreement, and have evaluated these mechanisms with respect to our performance measures. That some of our results have been unexpected indicates how much further we need to pursue our understanding of this complex topic. The development of a mathematical theory of convention evolution — suggested in [7], and tentatively begun by us in section 2 — is an obvious requirement. Other areas for future work include a further investigation of different strategy update functions (see sections 3.1 and 3.2), and more detailed investigation of the role played by social structure and communication (cf. [5]).

References

[1] R. Conte and C. Castelfranchi. Simulative understanding of norm functionalities in social groups. In *Simulating Societies-93: Pre-proceedings of the 1993 International Symposium on Approaches to Simulating Social Phenomena and Social Processes,* Certosa di Pontignano, Siena, Italy, July 1993.

[2] N. Findler and R. Malyankar. Alliances and social norms in societies of non-homogenous, interacting agents. In *Simulating Societies-93: Pre-proceedings of the 1993 International Symposium on Approaches to Simulating Social Phenomena and Social Processes,* Certosa di Pontignano, Siena, Italy, July 1993.

[3] C. V. Goldman and J. S. Rosenschein. Emergent coordination through the use of cooperative state-changing rules. In *Proceedings of the Twelfth International Workshop on Distributed Artificial Intelligence (IWDAI-93)*, pages 171–186, Hidden Valley, PA, May 1993.

[4] N. R. Jennings. Commitments and conventions: The foundation of coordination in multi-agent systems. *Knowledge Engineering Review,* 8(3):223–250, 1993.

[5] J. E. Kittock. Emergent conventions and the structure of multi-agent systems. In *Proceedings of the 1993 Santa Fe Institute Complex Systems Summer School,* 1993.

[6] D. Lewis. *Convention — A Philosophical Study*. Harvard University Press: Cambridge, MA, 1969.

[7] Y. Shoham and M. Tennenholtz. Emergent conventions in multi-agent systems. In C. Rich, W. Swartout, and B. Nebel, editors, *Proceedings of Knowledge Representation and Reasoning (KR&R-92)*, pages 225–231, 1992.

[8] Y. Shoham and M. Tennenholtz. On the synthesis of useful social laws for artificial agent societies. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, San Diego, CA, 1992.