

Simulating Computational Societies

Lloyd Kamara, Alexander Artikis, Brendan Neville and Jeremy Pitt

Imperial College of Science, Technology and Medicine, London, UK
Electrical Engineering Department, SW7 2BT
`l.kamara,a.artikis,brendan.neville,j.pitt@ic.ac.uk`

Abstract. Multi-agent systems can be considered from a variety of perspectives. One such perspective arises from considering the architecture of an agent itself. Another is that of an instantiated agent architecture and its interaction with its peers in a MAS. A third perspective is that of an external observer. These three perspectives cover a potentially overlapping but essentially distinct set of issues concerning MAS simulation and modelling. In this paper, we consider each of these perspectives in turn and demonstrate how a simulation framework can support a collective treatment of such concepts. We discuss the implications for agent development and agent society design arising from the results and analysis of our simulation approach.

1 Introduction

Multi-agent systems (MAS) can be considered from a variety of perspectives. One such perspective arises from considering the architecture of an agent itself (e.g. [18]). This is relevant to agent designers and implementors, as it concerns the *internal operation* of an agent. The basic architectural properties largely determine how the agent will interact with its environment. Another perspective to consider is that of an instantiated agent architecture and its interaction with its peers in a MAS. This covers both *communicative* and *socio-cognitive* aspects of agent interaction (e.g. [6,9,16]). In addition to being relevant to agent designers and implementors, this can be used to model, analyse and explain the behaviour of the agents in terms of *social theories* [6,9]. A third perspective is that of an external observer (e.g. [1,2,8]). Under such circumstances, we adopt a bird's eye view of computational systems — we are not concerned with the internal architecture of the participating agents. We can specify the legal and social aspects of these systems, such as the *normative positions* and *institutional powers* of the agents, without making any assumptions about mentalistic concepts like beliefs, desires or intentions.

Each of above perspectives concerns *interactive* aspects of agent systems and societies [17]. On activation, an *agent architecture* is expected to interact with its environment. The introduction of *socio-cognitive elements* to agent reasoning allows interactions to be characterised in anthropomorphic terms, with corresponding implications for agent ownership, control and responsibility [17,20]. *External observation* offers an abstract view of interactions, making it possible

to reason solely about the effects such interactions have on institutional aspects of agent systems.

We aim to enable society designers and agent developers to view simulated societies both at a micro (i.e. from each agent perspective) and a macro (i.e. from an external perspective) level (as is the case with *Gaia* [27]), with particular focus on the linking concept of agent interaction. To this end, we propose simulation tools which address concerns and requirements identified at both (and intermediate) levels. We refer to these tools collectively as our *simulation framework*. Pitt *et al.* [15] describe an abstract producer/consumer (APC) scenario where producers sell information to consumers. In this scenario the producers are explorer agents that map out the distribution of oil in their environment and consumers are cartographer agents that initiate contract-net protocols [23] (CNP) to acquire the maps from the explorers. The CNP is a generic, high-level protocol commonly used in (distributed) Artificial Intelligence for distributed problem solving, task allocation, or certain types of auction. At the highest level of abstraction, the CNP assumes a pool of nodes (agents). Each node controls resources and can perform tasks (offers services), and can interact directly with every other node. A node may require certain resources, and certain combinations of tasks to be performed, in order to solve a problem. That node will use the CNP to find another node or nodes that will perform tasks or provide resources that solves its problem. We have used variants of the CNP in Section 3 and Section 4 as a vehicle for analysing trust, experience and reputation from the socio-cognitive (micro) perspective; and power, permission and obligation from the external (macro) perspective. In keeping with our general approach, note that the protocol itself abstracts away from all such issues as the actual physical architecture of the individual nodes, the communications medium, interoperability, and so on.

The rest of this paper is organised as follows. In section 2, we present MAS and agent simulation architectures. In section 3, we describe an approach to trust and reputation modelling in MAS. In section 4, we consider executable specifications of norm-governed computational societies in a manner independent of agent internals. In section 5, we examine the results and implications for agent development and agent society design, concluding that they offer some preliminary guidelines to agent implementors and society developers for scoping *social* MAS.

2 Agent and MAS Simulation Architectures

The basis of our MAS simulation architecture is a collection of agents whose interaction is governed by a *communications interface* and an accompanying set of *protocols*. The MAS is neutral with respect to the architecture of participating agents: no assumptions are made about internal operation and motivations of an agent by its peers. Interaction between two agents takes place through their communications interfaces, either creating or extending a communicative context, or *conversation*.

2.1 Communications Interface

The communications interface consists of a higher and lower component pair: the *Agent Communication Language* (ACL) and a socket-based module for TCP/IP transmission of messages. The former defines (amongst other things) the supported syntax of messages while the latter defines how messages are exchanged. Following [16], we define an ACL in terms of three components: a *content language*, which defines the syntax of messages, a set of *protocols* which define patterns of interaction and a *reply function* which provides an external semantics for the ACL. The content language is the set of performatives that agents use in communication, while the protocols identify sequences of performatives that constitute meta-level interaction (such as auction and query/response protocols). Given an input performative, protocol and current conversation state, the reply function returns the set of ‘acceptable’ responses. These take the form of performative-protocol pairs, which, depending on the options available, may enable a responding agent to continue the conversation using the same protocol, or to initiate an auxiliary conversation using a different protocol. The relationship between content language, protocols and reply function is specified in an separate (**Prolog**) module, making it straightforward to refine and enhance the communicative behaviour of agents.

Pitt and Mamdani [16] justify the use of a protocol-based semantics to cover the external aspects of agent interaction in MAS featuring agents with heterogeneous architectures and distinct reasoning and motivational traits. A similar justification is used here, with the introduction of a socio-cognitive perspective arguably serving to re-inforce the need for such an approach. The additional subjective treatment of concepts including trust and reputation makes each agent even more distinct and thus external characterisations of interactions become increasingly relevant. In the following section, we describe an agent architecture that can be parameterised with different reasoning and socio-cognitive behaviour, allowing us to represent heterogeneous MAS in our experiments while retaining a compact simulation base.

2.2 Agent Architecture

In our proposed experimental configuration, a generic BDI [18] architecture forms the computational base for each agent. We use the same architecture for reasons of simplicity: the principle of heterogeneity outlined earlier has not been abandoned. It would be equally possible to use agents with different architectures as long as they possess compatible interfaces (like in [11, 13]).

Specific behaviour is determined by parameterisation of the respective agent instances. This approach can be used, for example, to pre-assign the roles of auctioneer and bidders in a group of agents enacting an electronic auction. Figure 1 gives a diagrammatic overview of the agent architecture.

The control module contains the agent interface and interpreter, in addition to housing the protocols sub-module. The agent interface facilitates and manages conversations (as mentioned previously), representations of which are to be found in the conversational component of the agent’s mental state.

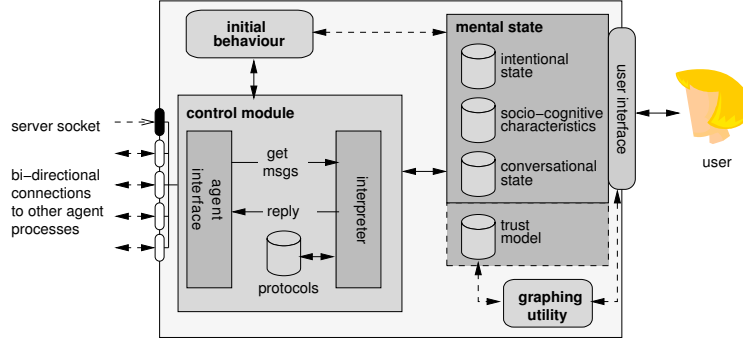


Fig. 1. Generic Agent Architecture for Experiments

The interpreter serves as the animating core of the agent architecture, operating in the form of a typical BDI cycle. It processes incoming messages at the content level, consulting and updating mental state representation while executing communicative actions appropriate to the agent’s recent perception and interpretation of events.

The mental state representation consists of several elements (see Figure 1). The intentional state holds the agent’s belief and motivational state alongside the agent’s current intentions. These are expressed as time-stamped **Prolog** terms with assertion being the means of updating state content. Records of conversations — again, represented through **Prolog** terms — are maintained in the conversational state component. Each conversation is recorded as a four-tuple denoting the correspondent (the other agent in the conversation), local and remote conversation identifiers [16] and the current state of the conversation (from a local perspective). A sub-set of the contained data reflects proceedings in current, or active, conversations. The user interface provides a means for the experimenter to initialise the agent and to monitor and control its behaviour.

The architecture does not feature a fixed procedure for belief update at present, although the way has been prepared for the introduction of such a mechanism. In particular, with each **Prolog** term used to represent a belief there is an associated *credence* measure (in the range of 0–1) which reflects the level of certainty the agent has in the content.

3 Socio-Cognitive Modelling

Our investigation of socially motivated behaviour is based on formal models of anthropomorphic socio-cognitive relations. Accordingly, our socio-cognitive model defines three component beliefs of each agent about its peers in the society: *trust*, *direct experience* and *reputation*. This section defines the socio-cognitive model and the software developed to facilitate systematic experimentation with agent societies whose members are implemented with a (parameterised) version of the model.

Our computational representation of *trust* is based on the formal model of Castelfranchi and Falcone [6,9]. The essential conceptualisation is as follows: the degree to which Agent A trusts Agent B about task τ in Ω (state of the world) is a subjective probability; this is the basis of agent A’s decision to rely upon B for τ . Our method extends this by defining trust as the resultant belief of one agent about another, borne out of direct experience of that other party and/or from the testimonies of peers (i.e. reputation).

We define *direct experience* as the belief one agent has about the trustworthiness of another, based on first-hand interactions. Having ‘trusted’ another agent to perform task τ and assessed the outcome, an agent will update its direct experience beliefs concerning the delegated agent accordingly. The outcome of delegating a task may be either successful or unsuccessful: this categorisation is the basis of an experience update rule [25] that calculates the revised level of trustworthiness to associate with the agent in question.

We briefly address a proposed method of combining direct experience with reputation to formulate the mental state of trust. Each belief has associated with it a degree of confidence signifying an agent’s trust in the belief’s accuracy. This confidence measure is essential to what trust will be based on, be it one or indeed both of the beliefs, experience or reputation. An agent with strong confidence in its experience beliefs and little confidence in the accuracy of its reputation beliefs should rationally choose to calculate its *degree of trust* (**DoT**) primarily from its experiences. An agent recently introduced to the system should have little confidence in its own (in)experience and should thus base its trust solely on reputation. The number of direct experiences is therefore significant in evaluating confidence in a belief about direct experience. It is similarly the case for reputation, where the number of agreeing testimonies is a decisive factor. Experience and reputation thus become influences of trust, the weighting assigned to them dependent upon an agent’s confidence in their respective accuracies.

Our motivation in investigating *reputation mechanisms* by simulation is provided by Conte [7], who outlines the need for ‘decentralised mechanisms of enforcement of social order’ noting that ‘reputation plays a crucial role in distributed social control’. In an agent society, reputation is the collectively informed opinion held by a group of agents about the performance of a peer agent within a specific social context. By consulting its peers, an agent can discover the individual reputation of an agent. However, the received testimonies may be affected by existing relationships and attitudes (for example, agent *C* may be willing to divulge reputation information to agent *A* but not agent *B*). Thus, reputation is also a subjective concept which we define as a belief held/derived by one agent.

The *Subjective Reputation Evaluation Function* (**SREF**) formulates subjectively — that is, from the perspective of an agent *A* — the reputation of an agent *B*, based on information from the peers of *A* and *B*. **SREF** may take several forms; examples are a weighted sum or a fuzzy relation. In our current work, we use the former approach; a summing function in which n assertions received from peers regarding agent *B* are weighted by the agent’s trust (confidence) in

each peer to make accurate recommendations. This approach incorporates the credibility of each assertion source (the credibility of a belief being dependent upon the credibility of its sources, evidences and supports [6]).

3.1 The Trading Economy

We simulate our commerce society as trading agents executing a variant of the CNP. In this version, the corresponding communicative acts for informing of the successful completion of a task and making payment are sent without any intermediate evaluation — that is, the cartographer (consumer) does not know how well (if at all) a task has been performed before it sends a payment in response to a contracted explorer’s (producer) *task completed* message. In effect, we have a contract-net concluding with a *prisoner’s dilemma* [3] (PD). Each agent’s actions at any stage will be influenced by expectations about actions of its trading counterpart. The agents are able to execute the CNP repeatedly, forming the basis for an iterated PD. This differs from the game theoretic studies in [3], however, in that agents choose the peers with which they are willing to enter the PD on a trust/cost trade-off basis. The reasoning behind which action to take once a contract is established is similarly distinguished.

By using the CNP, we introduce the concept of bidding, the explorer agent being able to vary bidding price. This could be used to increase the desirability of an agent’s bid, by under-cutting the bids of other explorers held in higher esteem. Agents not being awarded contracts may lower their prices in an attempt to generate business. Once a loyal trade relationship has been established, an explorer agent can gradually increase its bidding price in order to increase profitability.

When a cartographer awards a contract to a specific explorer, a PD-style dependence is obtained. The explorer is relying upon the cartographer to make prompt payment; the cartographer is relying upon the explorer to provide the required information. Our system however, does not guarantee that agent actions will always have the intended effect. Agents may fail to complete a task for reasons other than conscious deception. The inclusion of this *fallibility factor* is to account for unpredictability and unreliability within MAS environments and real-world applications.

3.2 Trust-based Prediction

Here, we discuss the use of a trust mechanism as a predictor for future peer behaviour based on past experiences and reputation. In this context, we interpret trust as a subjective probability that a peer will take a certain action, the outcome of which will characterise the corresponding interaction context. For each outcome, the agent will experience profit or loss to a varying extent — we refer to this as the outcome’s *pay-off*. Knowing the pay-offs of each outcome in advance and having an estimate of their probabilities of occurrence, the agent can calculate the expected pay-off of each of its options.

How the agent determines trading partners and what action to take (*cooperate* or *defect*) once a contract is established depends upon its character type and

the expected pay-offs of each possible action. We currently simulate three basic character types. *Co-operators* only choose cooperation strategies; if the expected outcome for cooperation is not greater than zero, a co-operator will not enter into a contract. *Defectors* will pursue a policy of cooperation or defection purely based upon maximising the expected pay-off of an interaction. *Reciprocators* have probabilistic intentions; the probability that they will co-operate (their *trustworthiness*) is matched to their evaluation of a trading partner’s behaviour (**DoT** in the peer). As a result, their trustworthiness displayed towards co-operators is high, whereas they are likely to defect against defectors.

3.3 Agent Configuration and Monitoring

The *Multiple Agent Launcher* (MAL) is a software tool that allows experimenters to create and launch an arbitrary number of agents with distinct configuration parameters. Agents may be launched individually or in groups. The configuration parameters are expressed as arguments to the agent architecture presented in section 2 and range from low-level (e.g. agent cycle delay time) to high-level (e.g. file-name of **Prolog** source for run-time behaviour). The motivation behind the MAL is to provide the user with a single interface from which to configure and monitor agent experiments, as well as to visualise real-time agent interactions. It is important to note that the agents thus launched remain semi-autonomous processes; the MAL is not a centralised control mechanism, but behaves as a convenient abstraction of one. The MAL supports configuration of agents with heterogeneous socio-cognitive characteristics and behavioural traits, by allowing the experimenter to adjust the parameters of the trust and reputation update functions (for example). The output of these functions is logged at various intervals during a simulation run and can also be graphed in real-time.

3.4 Simulation and Results

We now discuss an example simulation for trust update based on direct experience. The experiment consists of twelve agents, split evenly between producers (explorers) and consumers (cartographers). All of the agents are of reciprocator interaction type, differing only in their ability to complete the tasks relied upon by their respective contract partners. This ability is manifested through a probability parameter which determines the rate of successful task completion per agent. In this experiment, three ability levels were specified: 95%, 75% and 50%. Two cartographers and two explorers were all assigned the same probability parameter for each of the three levels, resulting in a tripartite ability configuration. When an agent fails to successfully perform its part of the contract, its partner perceives this and consequently reacts as if it were a conscious act of defection. In the graphs of Figure 2 and Figure 3, each line reflects the mean data values belonging to a pair of equally skilled agents of the same type. The mean data readings for two explorer agents whose degree of ability is 95%, for example, is labelled ‘Explorers (95%)’. Timepoints occur at 10 second intervals; the current trust beliefs and monetary worth of the agents are logged at each point.

Figure 2 plots the level of trust directed towards different members, over the time of the simulation. The featured **DoT** is simply the mean of the opinions of all the members in the society who know the agent in question. Agents are configured to initially blindly trust their peers (**DoT** = 1.0). As the simulation progresses their opinions change dramatically — in particular, there is a drop in communal trust for the capable agents. This is because the incompetent agents do not trust the competent agents, while the capable agents reciprocate for the incompetents’ failures by defecting against them. The competent agents have a very strong trust relationship among themselves but a very poor one with others — the average is therefore lowered. Our next graph (Figure 3) shows the per-

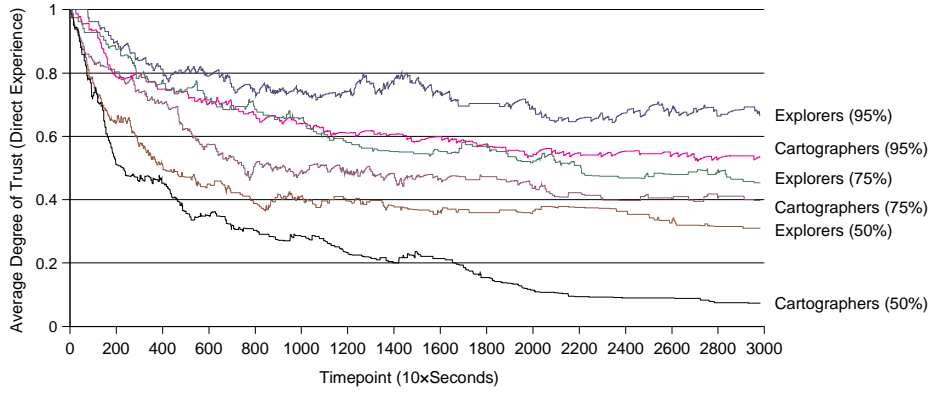


Fig. 2. Experimental Results - Trust Evolution

formance measure, the amount of assets accrued or lost. From this it can be seen that a society of reciprocators is a meritorious one. The highly capable agents (95%) are able to succeed; agents of intermediate skill (75%) break roughly even while the least skilled agents (50%) are punished.

4 Executing the Specifications of Computational Societies

Artikis *et al.* [1] present a theoretical framework for providing executable specifications of particular kinds of multi-agent systems, called *open computational societies*. Open societies (as defined in [1]) have the following characteristics: first, the internal architecture of the agents is not publicly known, i.e. there is no direct access to an agent’s mental state. Second, the members of the society do not necessarily have a common notion of global utility. Members may fail to, or choose not to, conform to specifications in order to achieve their individual goals. In addition to these properties, in open societies the behaviour of the members and their interactions cannot be predicted in advance. In this framework the computational systems are viewed from an external perspective. Three key components of computational systems are specified, namely the *social constraints*, *social roles* and *social states*. The specification of these concepts is based on and motivated by the formal study of legal and social systems (a

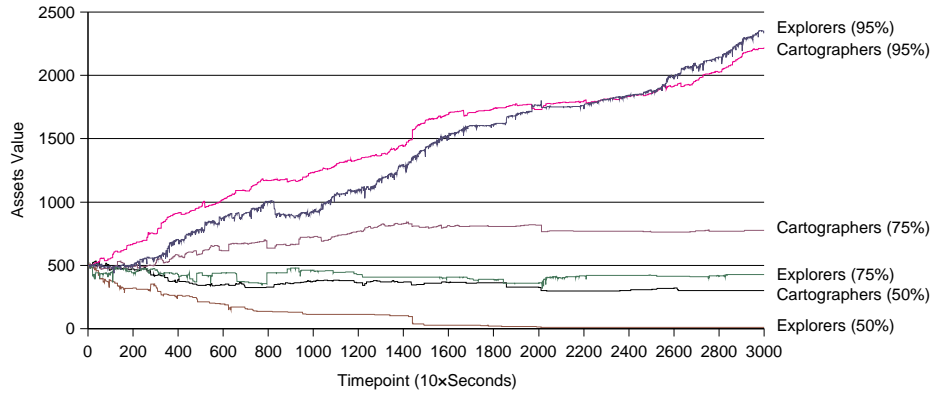


Fig. 3. Experimental Results - Monetary Performance

theory of institutionalised power [14], normative positions [21]) and traditional distributed computing techniques (state transition systems [8]). The social constraints and roles have been specified with two different formalisms from AI: the Event Calculus [22] (see [1]) and the *C+* language [12] (see [2]). Next, we briefly discuss the way social constraints are specified in [1, 2]. Then we present a computational framework that executes such specifications (i.e. the social constraints of the agent societies).

4.1 Social Constraints

Artikis *et al.* [1, 2] follow Jones and Sergot [14] and maintain the standard (in the study of social and legal systems) long established distinction between *permission*, *physical capability* and *institutionalised power*. The social constraints specify:

- What kind of actions ‘count as’ [14] *valid* (‘effective’) actions. Distinguishing between *valid* and *invalid actions* enables the separation of meaningful from meaningless activities.
- What kind of actions (valid, invalid) are *permitted*. Determining the *permitted*, *prohibited* and *obligatory actions* enables the classification of the agent behaviour as ‘legal’ or ‘illegal’, ‘social’ or ‘anti-social’, etc.
- What are the *sanctions* and *enforcement policies* that deal with ‘illegal’ or ‘anti-social’ behaviour.

Valid actions are specified as follows: An action counts as [14] a *valid action* at a point in time if and only if the agent that performed that action had the *institutionalised power* [14] to perform it at that point in time. Differentiating between valid (‘meaningful’) and invalid (‘meaningless’) actions is of great importance in the analysis of agent systems. For example, in an auction, the auctioneer has to determine which bids are valid and, therefore, which bids are eligible for winning the auction.

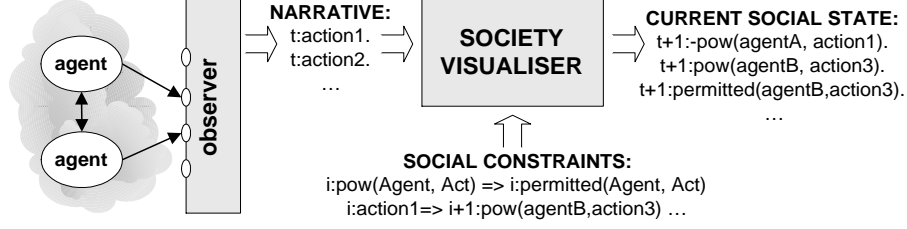


Fig. 4. The Society Visualiser.

4.2 The Society Visualiser

We describe a software platform called *Society Visualiser* (SV), that builds upon the theoretical framework (previously described) for the specification of open systems [1] and that compiles (produces) and updates the global state of an agent society. The global state includes information like the institutional powers, permissions, obligations, sanctions and social roles of the members. Figure 4 describes the architecture of the SV in terms of inputs/outputs. In order to produce the global state of the societies, the SV has two inputs: a *narrative* and the *social constraints*¹. The narrative is a description of the externally observable events that take place in a computational society. For example, $t : action1$ states that *action1* took place at time point t (Figure 4). The narrative is produced in the following manner: When a member of the society sends a message to a peer, he also sends that message to an *observer agent* for monitoring purposes². These messages are called *report* messages and are necessary for the compilation of the social states. The report messages (and all other monitored events) constitute the *narrative*. In other words, the SV observes (without intervening) the interactions of the members of the computational societies in order to produce the social states of the societies.

The social constraints specify the valid actions, institutional powers, permissions etc. of the members of the society. Consider the following example of a constraint (expressed in a generic notation):

$$i : pow(Agent, Act) \Rightarrow i : permitted(Agent, Act)$$

This constraint states that if at time point i an agent is ‘empowered’ [14] to perform an action (represented by $pow(Agent, Act)$) then this agent is also ‘permitted’ to perform the same action (represented by $permitted(Agent, Act)$). The remaining constraints are specified in a similar fashion.

¹ The social constraints and the narrative are expressed in terms of either the Event Calculus or the *C+* language. However, in order to simplify our analysis, we illustrate the social constraints and narrative in Figure 4 with the use of a generic notation.

² The observer agent monitors the interactions of the members for the benefit of the Society Visualiser.

As mentioned earlier, the output of the SV is the global state of the society at a point in time. In other words, given the narrative of time point t (say), the SV will produce the states of affairs that hold at time $t + 1$. Global states are stored in a database and are displayed in a graphical interface for the benefit of the society designer. In this graphical display the social state of each agent is divided in five categories: (i) institutional powers, (ii) permissions, (iii) obligations, (iv) sanctions and (v) valid actions. Figure 5 demonstrates the GUI of the SV during the simulations of a variation of the CNP (see [1] for more details). Global states are produced for the benefit of the members of the societies as well. Agents can send *query* messages to the SV in order to find out the social state of the group or of their peers. Apart from producing the global state of each time point,

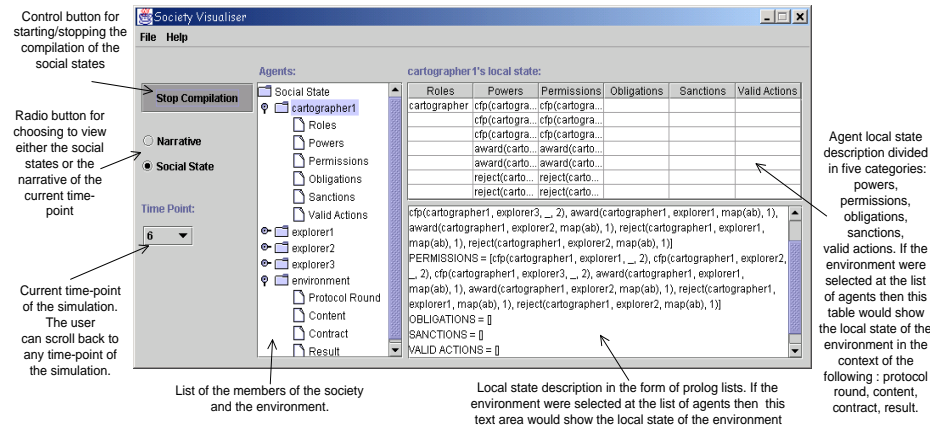


Fig. 5. The main GUI of the SV.

the SV includes a number of additional capabilities, one of which is *planning*. Given an initial state and a goal state (and the set of social constraints), the SV can determine if there exists a sequence of actions that will lead from the initial to the goal state. Due to the high complexity of such tasks, computation of planning queries can be mainly performed in an *off-line phase*, that is before the commencement of the interactions of the agent societies. Such a functionality enables the society designer to prove various properties of the agent systems and, therefore, evaluate the design of the social constraints. For example, having specified the constraints of an auction protocol, the designer can determine with the use of planning queries if it is possible to reach a state where two different agents have won the auction. Currently, this functionality is only provided if the social constraints are expressed by means of the *C+* language³.

5 Summary and Conclusions

We presented a simulation framework that addresses the internal architecture of the agents in addition to accounts of the trust relationships of the agents and of

³ In this case the SV is an extended version of a software tool called **Causal Calculator** (see [2]).

the institutional and social aspects of agent systems. We intentionally did not address/fully investigate issues like low-level details of the software components, experimentation, scalability and other criteria discussed in the community for the experimental study of MAS (see for example [10,11]). Our aim is to stress the need for the representation of conceptually different perspectives in the modelling and simulation of computational societies. The issues that were not addressed in this paper will be investigated in future work⁴.

Several simulation platforms have similar objectives to the framework presented in this paper (e.g. MACE3J [11], MadKit [13], FishMarket [19] and MYWORLD [26]). To our knowledge, these platforms do not formally address all issues raised in this paper. For example, in Fishmarket there is a lack of formalisation of concepts such as rights, obligations and social relations as well as an explicit representation of them during simulations. The MadKit platform is based on the organisation metaphor AGR (agent/group/role) developed in the context of the AALAADIN project and integrates heterogeneous agent systems. As pointed out in [29], the AGR model views organisations simply as collections of roles and does not incorporate the necessary notion of organisational rules (i.e. social constraints).

The analysis and interpretation of the results from our trust simulations is ongoing work, but we believe some conclusions can already be reached. There is every indication that the use of trust-based prediction accurately reflects the characterisation of agent capabilities in experiments. From this we infer that building a socio-technical layer of computation, complementary to (and exploiting functionality of) a lower level of security, is technically feasible, and indeed offers a more tractable solution to certain security problems in e-commerce [4,24] and digital rights management than, say, ever more sophisticated encryption algorithms or encoding technology. For example, the proponents of the Extensible rights Mark-up Language (XrML) [28] propose XrML as the foundation of trusted system development: we propose *trust* as the foundation of trusted system development

Although our presented experiments have not covered reputation, we intend to address this in future work, as we believe it is an important aspect of agent interaction. In particular, we note how reputation is related to the notion of scalability: in a system with a large number of agents, the likelihood of any individual agent having interacted with a specific peer is less. This leads to an overall reduction in the amount of direct experience available to an agent during its decision-making. We are investigating when and how agents (in their capacity as third parties for reputation information transmission) can effectively propagate information [7] under such circumstances. An important aspect of future work will be trying to establish what parameters and socio-cognitive characteristics lead to ‘stable’ trust relationships; ones that are resistant to minor aberrations in agent behaviour due to circumstances beyond their control.

⁴ Limited space is another reason why we do not fully investigate these issues here. However, details on issues like experimental results can be found in [1] and [2].

There exist several approaches that modify the traditional BDI cycle in order to increase the ‘social awareness’ of agents (e.g. [5]). The architecture we have presented does not reason about deontic concepts, but one of our objectives is to make the information compiled by the Society Visualizer available to other agents, either through communication (via *query* messages) or via the addition of an equivalent module to each agent. Reasoning about the institutional and social properties of the agents (as produced by the Society Visualiser) is also a crucial part of trust relationships. Members of a society, governed by a set of rules, must consider when deciding to delegate a task to a peer what are its associated institutional powers, permissions, obligations, sanctions and social roles. For example, it would not be ‘rational’ to delegate a task to an agent if that agent does not have the institutional power or is forbidden to perform that task. Future work includes modifying the process of the trust update mechanism by considering attributes such as powers (institutional and physical), permissions, obligations and roles of the peers.

In conclusion, we believe the combination of architectural, socio-cognitive and external observational perspectives presented in this paper affords a comprehensive approach to modelling MAS. We plan to fully validate the presented argument through continued simulation and analysis.

6 Acknowledgements

This work has been undertaken in the context of the EU-funded ALFEBIITE Project (IST-1999-10298). We have also benefitted from Marek Sergot’s contributions on the specification of agent societies (Section 4).

References

1. A. Artikis, J. Pitt, and M. Sergot. Animated specifications of computational societies. In C. Castelfranchi and L. Johnson, editors, *Proceedings of Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 1053–1062, 2002.
2. A. Artikis, M. Sergot, and J. Pitt. Specifying electronic societies with the causal calculator. In *Proceedings of the Third International Workshop on Agent-Oriented Software Engineering (AOSE-2002)*, pages 75–86, 2002.
3. R. Axelrod. *The Evolution of Cooperation*. Basic Books, New York, 1984.
4. M. Blaze, J. Feigenbaum, John Ioannidis, and Angelo D. Keromytis. The role of trust management in distributed systems security. In *Secure Internet Programming: Security Issues for Mobile and Distributed Objects*, number 1603 in Lecture Notes in Computer Science, pages 185–210. Springer, 1999.
5. J. Broersen, M. Dastani, Z. Huang, J. Hulstijn, and L. Van der Torre. The BOID architecture: Conflicts between beliefs, obligations, intentions and desires. In *Proceedings of Autonomous Agents*, pages 9–16. ACM Press, 2001.
6. C. Castelfranchi and R. Falcone. Social trust: A cognitive approach. In *Trust and Deception in Virtual Societies*, pages 55–90. Kluwer Academic Press, 2000.
7. R. Conte. A cognitive memetic analysis of reputation. *Alfebiite* project deliverable, <http://alfebiite.ee.ic.ac.uk/docs/Deliverables/D5D6.zip>, 2002.
8. R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning About Knowledge*. The MIT Press, 1995.

9. R. Falcone and C. Castelfranchi. The socio-cognitive dynamics of trust: Does trust create trust? In *Trust in Cyber-Societies*, volume 2246 of *LNAI*, 2001.
10. L. Gasser. MAS infrastructure definitions, needs, prospects. In *Infrastructure for Agents, Multi-Agent Systems, and Scalable Multi-Agent Systems*, 2001.
11. L. Gasser and K. Kakugawa. MACE3J: Fast flexible distributed simulation of large, large-grain multi-agent systems. In C. Castelfranchi and L. Johnson, editors, *Proceedings of Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 745–852, 2002.
12. E. Giunchiglia, J. Lee, V. Lifschitz, N. McCain, and H. Turner. Nonmonotonic causal theories. To be published, 2001. <http://www.d.umn.edu/~hudson/nmct.ps>.
13. O. Gutknecht, J. Ferber, and F. Michel. Integrating tools and infrastructures for generic multi-agent systems. In *Proceedings of Autonomous Agents*, 2001.
14. A. Jones and M. Sergot. A formal characterisation of institutionalised power. *Journal of the IGPL*, 4(3), 1996.
15. J. Pitt, L. Kamara, and A. Artikis. Interaction patterns and observable commitments in multi-agent trading scenario. In *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 481–489, 2001.
16. J. Pitt and A. Mamdani. Designing agent communication languages for multi-agent systems. In *Multi-Agent System Engineering: Proceedings of MAAMAW'99*, volume 1647 of *LNAI*, pages 102–114. Springer-Verlag, 1999.
17. J. Pitt, A. Mamdani, and P. Charlton. The open agent society and its enemies: A position statement and research program. *Telematics and Informatics*, pages 67–87, 2001.
18. A. Rao and M. Georgeff. BDI agents: from theory to practice. In Victor Lesser, editor, *Proceedings of ICMAS*, pages 312–319, San Francisco, CA, 1995. MIT Press.
19. J. Rodriguez-Aguilar, F. Martin, P. Noriega, P. Garcia, and C. Sierra. Towards a test-bed for trading agents in electronic auction markets. In *AI Communications*, pages 5–19, 1998.
20. Giovanni Sartor, editor. *Proceedings of the First Workshop on The Law of Electronic Agents (LEA)*. 2002.
21. M. Sergot. A computational theory of normative positions. *ACM Transactions on Computational Logic*, 2001.
22. M. Shanahan. The event calculus explained. *Artificial Intelligence Today*, 1999.
23. R. Smith and R. Davis. Distributed problem solving: The contract-net approach. In *2nd Conference of Canadian Society for CSI*, 1978.
24. V. Swarup and J. T. Fabréga. Trust: Benefits, models and mechanisms. In *Secure Internet Programming: Security Issues for Mobile and Distributed Objects*, number 1603 in Lecture Notes in Computer Science, pages 3–18. Springer, 1999.
25. M. Witkowski, A. Artikis, and J. Pitt. *Experiments in Building Experiential Trust in a Society of Objective-Trust Based Agents*, pages 111–133. Number 2246 in *LNAI*. Springer-Verlag, 2001.
26. M. Wooldridge. This is MYWORLD: The logic of an agent-oriented DAI testbed. In *Intelligent Agents: The 1994 Workshop on ATAL*. Springer-Verlag, 1995.
27. M. Wooldridge, N. Jennings, and D. Kinny. The Gaia methodology for agent-oriented analysis and design. *Journal of Autonomous Agents and Multi-Agent Systems*, 3(3):285–312, 2000.
28. XrML. Extensible rights mark-up language. <http://www.xrml.org>.
29. F. Zambonelli, N. Jennings, and M. Wooldridge. Organisational rules as an abstraction for the analysis and design of multi-agent systems. *International Journal of Software Engineering and Knowledge Engineering*, 11(3):303–328, 2001.