Social Minded Commitment Management

Robert Ross[1] and **Rem Collier**[2] and **G.M.P. O'Hare**[3]

Abstract. Over the past 30 years Artificial Intelligence has fragmented from one broad subject into a cluster of narrow but deep individual disciplines. During this time we have also seen the development of increasingly complex software systems for application domains such as robot control, mobile computing, and expert system interfaces. Many of these designs use elements from the branches of AI, but pay little attention to the integration of these elements in an intelligent way. This paper presents an approach to this intelligent integration problem, based on a community of Intentional Agents. Each of the agents within the community uses a Social Minded Commitment Manager (SMCM) to allow it to reason and cooperate in order to achieve goals when individual execution has failed. An implementation of the SMCM that has been developed for AgentFactory is presented, and its use then motivated through the description of a robust, redundancy tolerant robot control architecture named MARC.

1 Introduction

Over the past 30 years there has been a fragmentation of Artificial Intelligence into a multitude of deep specialised subdisciplines. Each of these individual branches are undoubtedly very important, providing us with useful algorithms for data extraction, natural language processing, planning and so forth. Little attention has however been given to the question of how individual components can be intelligently integrated in complex system designs.

The need for intelligent integration of AI techniques and algorithms is perhaps nowhere more manifest than in the production of intelligent service robots. Whereas simple control algorithms sufficed in the 80s, modern robot control architectures must integrate a diverse range of components that deliver support for tasks such as motor control, dialog management, and object recognition. Despite this large increase in the complexity of control systems, we have not seen a significant change in the approach taken to the integration of the individual components within these architectures.

Static, brittle, tightly-coupled architectures are still the order of the day in the large software designs. Although some level of disjunction is possible through the use of standards such as the Distributed Component Object Model (DCOM), these are merely communication protocols and do not improve the intelligence of the individual component or the larger system. The creation of large software systems using C like monolithic architectures, object oriented or DCOM models leads both to multiple points of critical failure, and a tightly coupled design which is not suitable for extension.

The authors reject this static design approach in favour of a dynamic metaphor based around a community of Intelligent Intentional Agents. In this community, agents are strong software entities which have inbuilt reasoning and plan execution ability. Such abilities allow for communication and cooperation within a larger disjoint software architecture. To improve this feasibility of this approach, we introduce a Social Minded Commitment Manager (SMCM) which improves on the basic cooperative skills of Intentional Agents. The SMCM is based around a formal Intentional Agent model. Details of this formal model are beyond the scope of this paper, and instead, the reader is directed to [2]. However, in order to facilitate the description of the SMCM and its implementation, we start with an informal review of the concept of an Intentional Agent.

2 Intelligent Intentional Agents

The notion of an Intentional Agent is broadly based on the work of the philosopher Daniel Dennett. In [4] Dennett introduces the concept of the "Intentional Stance" as a more appropriate way of modelling complex systems. Specifically, Dennett argues that, through the ascription of folk psychological notions such as beliefs, hopes, and goals, people are more easily able to understand behaviour of complex systems that, through the more traditional physical and design stances. It is this notion of the Intentional Stance, as applied from an internal perspective as a tool for modelling both the agent and its environment that categorises an Intentional Agent.

Initial work on Intentional Agents led to the design of a number of agent architectures that define the data structures and algorithms that are required to implement such an agent [5] [1]. For a review of some of the more prominent Intentional Agent architectures see [2]. Many of these architecture were based upon earlier theoretical work on Intentional Agents that employed three mental notions: Beliefs, Desires, and Intentions. Specifically, beliefs are taken to represent the agents current subjective knowledge of itself and its environment; desires represent the agents ideal state of the environment; and the intentions represent a chosen subset of those desires that the agent is committed to bringing about. Architectures that employ these mental notions, or variant of them, have become known as Belief-Desire-Intention (BDI) architectures. Similarly, theories based upon beliefs, desires and intentions have become known as BDI theories [6] [9] [11].

A chief concern underlying the area of Intentional Agents is the identification of a clear link between the various BDI theories and the associated BDI architectures. This issue

[1] Universitat Bremen, Germany, email: robertr@tzi.de
[2] University College Dublin, Ireland. email: rem.collier@ucd.ie
[3] University College Dublin, Ireland. email: gregory.ohare@ucd.ie

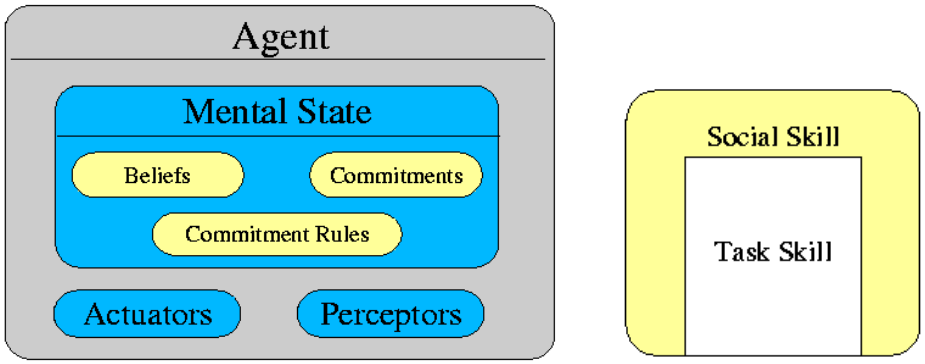

Figure 1. Two views of a Social Intentional Agent. On the left hand side we see an agent broken down in terms of its individual attributes e.g. beliefs, perceptors. On the right hand side we see an Intentional Agent as an entity with two distinct ability layers. This agent has high level reasoning and social abilities at the intentional level, while possessing task specific abilities to perform actions on data or the environment. An Intentional Agent Implementation provides the high level layer to all agents, while task specific abilities are 'plugged' into the core at application design time.

has been the subject of much research, and has led to the emergence of a class of programming language, known as Agent-Oriented Programming (AOP) languages [10]. These languages attempt to provide a strong link between the BDI theories, which are often logical and based upon the computationally intractable Possible Worlds semantics, and the syntax and semantics of the associated language. Some of the more prominent AOP languages include Agent-0 [10], AgentSpeak(L) [8], 3APL [3], and AF-APL [2].

At a high-level, these languages provide constructs for representing mental attitudes such as beliefs, goals, and commitments, together with sophisticated reasoning engines which relate and revise these mental attitudes. Conversely, at a low level the agents abilities are task related, and must be provided through application specific actuators and perceptors. Essentially the high level model provides control of what to do and why, while the low level provides a means of doing things (See figure 1).

A large number of BDI/Intentional Agent implementations centre around a formal model of commitment. Whereas classical systems reasoned about GOAL and ACTION, the intentional agent also reasons about the more abstract notion of commitments. Commitments may be understood by their common language meaning in that they are promises made by one agent to another (or oneself). By reasoning and managing these commitments, a more flexible approach to agent control is possible then would be available through the management of actions alone. Commitments between agents allow for a basic level of cooperation to take place between these agents. In terms of performing complex actions, the vast majority of Intentional Agent implementations acknowledge the need for an agent to use plans to achieve complex tasks. Unfortunately however many implementations choose to execute the plan directly. An alternative approach is to extended the commitment model to handle plan constructs directly. Collier [2] presented a commitment model and implementation where complex plans were resolved to constructs of commitments at runtime. Although increasing the complexity of the agents execution model, such an approach adds great amounts of flexibility over the direct plan execution approach.

In any Intentional Agent implementation, the management of commitments is one of the core processes within each agent. Furthermore the commitment management methodology is one of the most distinguishing features between different Intentional Agent formulations. Probably the best known variants on commitment management are concerned with the maintenence condition for a commitment i.e. under what conditions a commitment is adopted, maintained and dropped [9]. Although these commitment management approaches and their successors, recognise external agents as those who make requests for commitments, the external agents are ignored in the process of achieving commitments. The next section introduces a social commitment manager which although broadly based on the underlying logic of existing commitment managers, uses the agents social environment to help in the commitment management process in times of adversity.

3 Social Minded Commitment Management

A Social Minded Commitment Manager is now presented. The essential difference between this commitment manager and its predecessors is the inclusion of basic social skills in the commitment management cycle. Through the use of these skills the intent is to bridge a gap between tradition Intentional Agent based Agent Oriented Software Engineering, and the long promised emergent qualities of reactive Multi-Agent Systems.

3.1 The Approach

Commitment Managers have traditionally had a limited intelligence in how they attempt to manage and achieve their commitments. Although Collier's [2] run-time adoption of commitment structures to achieve plans was a substantial improvement over more standard direct plan execution, there is still little intelligence in failure handling. When an action within a plan fails then the agent/commitment manager can

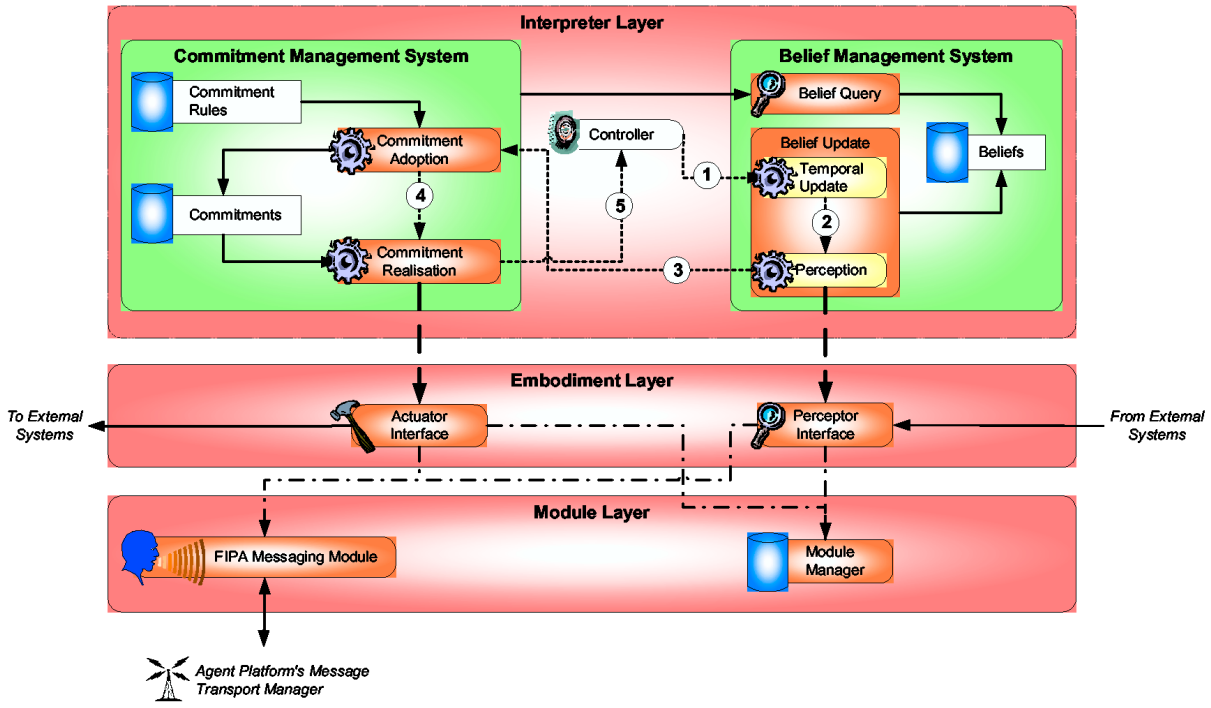

Figure 2. The AgentFactory Agent Programming Language Interpreter. The AF-APL comprises three layers. At the bottom a Module Layer for resource management, followed by an embodiment layer which governs perceptor and actuator execution. At the top layer, the Social Minded Commitment Manager sits alongside the Belief Management System, providing all high level reasoning for the agent.

only fall back on the contents of the plan. If the plan was not formulated in a way to deal with this situation explicitly, then the agent's commitments can fail easily.

Social Minded Commitment Management (SMCM) is different from other approaches in that the commitment manager is specifically designed to incorporate social skills to compensate for an agent's initial failure to achieve a commitment. To put it in other words, if an agent initially fails to achieve a task, then the SMCM is explicitly designed to try to achieve the task with the aid of *any* agents within it's environment. Rather than being an application specific ability, this behaviour is encoded within the intentional agent framework, thus endowing all such agents with these social abilities (See figure 1).

This loosely coupled connection used in failure handling is used at other times in SMCM based agents. For example acquaintances should not be hard-coded into design files or initialisation scripts. Instead all acquaintances are acquired at run-time with a dynamic *'Yellow Pages'*approach which allows an agent to initiate basic communication with any other agent in its environment. Whether communication continues, or is in any way productive is entirely dependant on the needs of individual agents. To improve openness this acquaintance acquisition process takes place on a regular basis, allowing newly inserted, copied or spawned agents to communicate and cooperate with the most appropriate agents within the community.

An agent using the SMCM can be seen in very abstract form in figure 1. The core of the agent are low level task capabilites. These capabilites are entirely application dependent and concern anything from sorting algorithms, to speech generation, to basic movement behaviours. Sitting on top of these task skills are the social and reasoning capabilites provided by the SMCM. Many of these qualities are provided by default by the SMCM and do not need to be designed for specific applications. These qualities of the social intention agent allow for the easy fabrication of communities of agents to perform tasks in the production of complex control systems. The loose coupling of this community of agents brings us close to the flexibility and robustness of a MAS, while the inherent social and reasoning skills allow for the creation of individually powerful agents capable of planning and deliberation. The implementation of the SMCM was based on a commitment management model originally given in [2]. The updated model is now discussed with emphasis on those features which were necessary to the production of the *social* aspects of the commitment manager.

3.2 Implementation - Extending AgentFactory

The Social Minded Commitment Manager approach discussed above is abstract in that it could be implemented on many different Intentional Agent Frameworks. In practice the system has already been implemented as an extension to the the

AgentFactory - Agent Prototyping Environment [4]. AgentFactory provides many other important constructs and resources needed to build intelligent intentional agents. These components include a *Belief Management* and a framework for the definition of plans, actuators and perceptors (see fig 2). AgentFactory also provides low level communication facilities which allow agents to communicate both on the same and multiple platforms. Communication is FIPA compliant, which means that AgentFactory agents can communicate in a meaningful way with FIPA compliant agents based on other systems.

The implementation of SMCM required a number of extensions to the AgentFactory development environment, and a reworking of the formal commitment model. Some changes were relatively trivial such as the introduction of mechanisms to allow an agent to become acquainted with all agents in it's environment at runtime. Other extensions included the expansion of the planning language to allow for universal operations over a set of elements within the agents mental state. Further extensions included the introduction of agent introspection, and the implementation of the Social Minded Commitment Manager algorithm. These two items will now be discussed in more detail.

Introspection Introspection essentially allows an agent to answer *what if* questions from another agent and from itself. More specifically an agent (Agent-A) can examine its own mental state to determine in advance the probable outcome of a request by another agent (Agent-B). Introspection is typically used when Agent-B makes an inquiry as to how the Agent-A would hypothetically respond to some request. Based on the results of its own introspection, Agent-A can then inform Agent-B of the possible result of the request (i.e. whether Agent-A would commit to the request or not). This information on the run-time capabilites of the agent, can then be used in the formulation of initial joint plans. Naturally either the agent or world state can change in between the initial introspection request and a subsequent actual request for the action. However the initial introspection result can often allow for the creation of plans which are successful in many cases.

Introspection is modelled as a core agent actuator which operates on the mental-state of the agent. During execution of this actuator a clone of the agent's mental state is made, and this mental state is run as if an actual request for GOAL from Agent-B had been received. The results of this hypothetical request to the agent can then be used to formulate a response to the hypothetical question from Agent-B.

The Social Minded Commitment Manager

The Social Minded Commitment Manager algorithm was built on-top of an improved plan description language, introspection, and the ability to dynamically acquire acquaintances within the environment. As mentioned earlier the implementation and model used are broadly based on that presented in [2]. In practice the implementation of the commitment manager is extremely complex, therefore a highly simplified view of it is presented in figure 3.

During any given execution cycle of an agent, a previously held commitment to some activity might be attempted by the agent. Traditionally an invalidation of the pre-condition or a problem with the direct execution of an actuator would cause a commitment to fail, and inevitably being dropped. Instead the SMCM commits the agent to a social plan to achieve the action through the help of other agents within the environment/platform. The typical structure of such a social plan is presented in figure 4.

[4] See www.agentfactory.com

```
manageCommitments()
{
   foreach(commitment_to_primitive)
   {
     // attempt to achieve commitment

     // if commitment fails due to invalid
     // pre-condtions on the actuator
     // or plan being attempted
          // then commit to a social plan
          // to get help from other agents
          // to achieve the goal.
   }
}
```

Figure 3. Simplistic view of the SMCM algorithm.

```
PLAN get_help(?goal);
BODY
  SEQ(acquire_acquantances,
      FOREACH(BELIEF(friend(?agent)),
           XOR(SEQ(request(capable(?goal),?agent)
               await_response(capable(?goal),?agent),
               request(?goal,?agent),
               await_response(complete(?goal),?agent),
               adopt_belief(?goal)   )   )   )
      );
```

Figure 4. A Social Plan which may be committed to in order to get help from another agent to achieve a goal ?goal automatically.

The plan is typical of a set of social plans used by the SMCM to achieve social goals. It is a social plan simply, in that it is a plan which is particularly concerned with social interaction. The plan is initialised with *?goal* which is some state of the world or action which must be achieved by an agent in order to facilitate the achievement of the original commitment. The first step of the plan involves an attempt to become acquainted with all agents contactable on the agent platform. This *acquire_acquaintances* is implemented by another plan which uses the agent platform to get a list of all agents which are interested in potentially giving aid to this agent. Each potential helper is then listed in Agent-A's mental state as a friend(?agent) where ?agent is a variable which resolves to a unique identifier of the friend agent.

The next step in the plan is a FOREACH term which operates over all of the friends which are held by the agent at that time. The second term of FOREACH will be expanded out for each ?agent which was resolved against friend(?agent). This section of the plan to be expanded is a XOR operation, which operates on a more basic plan segment which uses basic speech acts and introspection abilities to find one agent which

is capable of achieving ?goal for AGENT-A. If any agent is found, they will be requested to achieve ?goal, and if they report they were successful in that undertaking,then the ?goal will be added to the agents mental state. Such a successful outcome will then allow the agent to fulfil its original requirements.

This use of social ability to achieve commitments during failure conditions, is a unique feature of the Social Minded Commitment Manager. This is in contrast with other Commitment Managers which would give up on the commitment at that point, and instead resort to complete re-planning to achieving the high level goal. This in-built social skill allows a community of intentional agents operates more like a MAS, providing robustness through very loose coupling. A key point here is that these are basic skills which come out of the use of the commitment manager, and do not have to be explicitly considered by a designer in the process of fabricating individual agents. To demonstrate the SMCM approach, an application in the area of mobile robot control will now be described.

4 Application - Multi-Agent Robot Control

The SMCM has been successfully deployed in the field of autonomous robot control. The field of robot control architecture design has been a fruitful field of study for AI over the past 30 years, with an evolution of approaches to control. The first planning based *Sense-Plan-Act* architectures, gave way to the new school of reactive architectures in the mid 80s. Purely reactive architectures then gave way for the emergence of hybrid architectures in the early 90s. These hybrid architectures in principle combine the best parts of both the Sense-Plan-Act and reactive approaches.

Hybrid architectures are however not without fault, and hybrid architectures to date suffer from the deficiencies of static and monolithic design. Often the top layers of these systems are built around one *all powerful* agent [7]. This rigid methodology provides not only problems in initial integration, but also leeds to a lack of system robustness, since the failure of any one component can lead to a cascading failure of the entire system. One approach might be to rigidly model and formalise the design, to the extent that all behaviour can be explicitly predicted and analysed against requirements. However such an approach is unrealistic in systems using a vary large number of individual software components. A more dynamic approach to the construction of these systems is necessary. A loosely coupled intelligent integration framework can help to reduce many of these issues. Not only that but a loose MAS like coupling, leaves the door open for the emergent behaviour to meet a myriad of situations which were not pre-built into the system design.

To this end MARC the Multi-Agent Robot Control architecture was developed. MARC is shown in abstract form in figure 5. The architecture is a true hybrid architecture with functional, reactive, sequencing and planning capabilities. The architecture differs from other approaches though in that reactive, sequencing and planning capabilities are modelled as a community of social agents which vary in their deliberative and reactive capabilites. All agents within this community have been built using AgentFactory. Those not requiring reactive control use the full SMCM, while those requiring reactive support, forgo the full SMCM for a reactive commitment management model model. Essentially these reactive agents can answer requests from other agents, but are incapable of using SMCM mechanism for failure recovery.

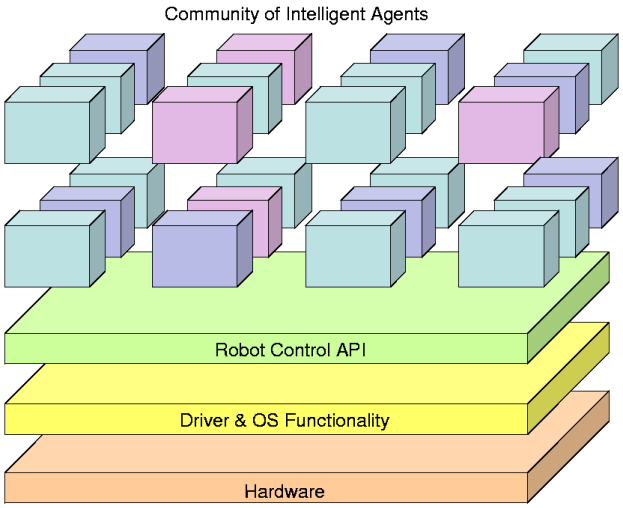

Figure 5. MARC - The Multi-Agent Robot Control architecture - Simplistic Layered View. The Social Minded Commitment Manager is used in the production of Deliberative Agents at the Community of Agents Level of Control.

MARC is being implemented as a control architecture for highly complex humanoid style robots. To this end many natural language processing agents have been developed in addition to the standard movement and command processing abilities normally associated with a mobile robot implementation. The architecture has been successfully deployed on Nomadic Scout II robots in University College Dublin, and is currently being deployed on the Rolland, the autonomous wheelchair in the University of Bremen, Germany.

5 Related Work

Haddadi has recently addressed the question of how intentional agents can form social relationships to achieve complex tasks [6]. Her formulation mainly focused on the production of basic plans through commitment negotiation. The approach did not however deal with the realities of failure and negotiation to recover from a failure which has already taken place.

6 Conclusions & Future Work

This paper presented the Social Minded Commitment Manager as an extension of traditional Intentional Agent Commitment Management approaches. The commitment manager uses social plans and run-time cooperation to attempt to maintain commitments in times of adversity. Such a design leads to a more dynamic Multi-Agent System based approach to complex software architectures, while maintaining the inherent computational power of Intentional Agents. The SMCM has been successfully integrated into AgentFactory,

and has been subsequently used in the production of a robust robot architecture. It is intended that the SMCM implementation brings us one step closer to the intelligent integration of complex software systems.

Specific future work on the SMCM includes the extension of dynamic planning capabilites available to the SMCM. With relation to this, extensions will be provided to allow for true joint planning based on introspection and dialog. Non development work on the SMCM includes the derivations of experimental scenarios which allow for its quantitative evaluation against more traditional commitment managers. This however is non-trivial since the SMCM is intended to be most useful in highly complex software architectures, which are inherently difficult to quantitatively evaluate.

6.0.1 Acknowledgements

We gratefully acknowledge the support of Enterprise Ireland through grant No. IF/2001/02, SAID.

REFERENCES

[1] M. Bratman, D.J. Israel, and M.E. Pollock. Plan and resource-bounded practical reasoning, 1998. Computational Intelligence 4(4), pp349-355,.

[2] Rem W. Collier, *Agent Factory: A Framework for hte Engineering of Agent Oriented Applications*, Ph.D. dissertation, University College Dublin, 2001.

[3] M. Dastani, F. Dignum, and J.J. Meyer. 3apl: A programming language for cognitive agents. ercim news, european research, 2000. Consortium for Informatics and Mathematics, Special issue on Cognitive Systems, No. 53,.

[4] Daniel C. Dennett, *The intentional stance*, The MIT Press, Massachusetts, 1987. 388 pages, 1987.

[5] M.P. Georgeff and F.F. Ingrand, 'Decision-making in an embedded reasoning system, proceedings of the international', in *Joint Conference on Artificial Intelligence (IJCAI'89), pp 202-206, Detroit, Michigan, USA, 1989*, (1989).

[6] Afsaneh Haddadi, *Communciation and Cooperation in Agent Systems: A Pragmatic Theory*, number 1056 in Lecture Notes in Computer Science, Springer-Verlag: Heidelberg, Germany, 1996.

[7] K. Konolige, K. L. Myers, E. H. Ruspini, and A. Saffiotti, 'The Saphira architecture: A design for autonomy', *Journal of experimental & theoretical artificial intelligence: JETAI*, **9**(1), 215–235, (1997).

[8] A. Rao. Agentspeak(l): Bdi agents speak out in a logical computable language, seventh european workshop on modelling, 1996. Autonomous Agents in a Multi-Agent World, Institute for Perception Research, Eindhoven, The Netherlands.

[9] Anand S. Rao and Michael P. Georgeff, 'Modeling rational agents within a BDI-architecture', in *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, eds., James Allen, Richard Fikes, and Erik Sandewall, pp. 473–484. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, (April 1991).

[10] Yoav Shoham, 'Agent oriented programming', *Artificial Intelligence*, **60**, 51–92, (1993).

[11] M. Wooldridge, *Reasoning about Rational Agents*, Intelligent Robots and Autonomous Agents, The MIT Press, Cambridge, Massachusetts, 2000.