**RAC**$\mathbb{SAM}$
*Rev. R. Acad. Cien. Serie A. Mat.*
VOL. **98** (1), 2004, pp. 77–83
Ciencias de la Computación / Computational Sciences

# Agent-Oriented Abstraction

**Jacques Calmet, Pierre Maret and Regine Endsuleit**

**Abstract.** We define an agent-oriented abstraction formalism devoted to generalized theories of abstraction that have been proposed in Artificial Intelligence. The model we propose extends the abstraction capabilities of the existing Agent-Oriented Programming paradigm. This short note reviews first the existing attempts to define abstraction in AI and in agent systems. Then, our model is introduced in terms of six definitions covering the concepts of agents, annotated knowledge, utility and society of agents. The presentation is purely cognitive and intuitive.

**Abstracción orientada a agentes**

**Resumen.** Definimos un formalismo de abstracción orientada a agentes dedicado a generalizar teorías de abstracción que han sido propuestas en Inteligencia Artificial. El modelo que proponemos amplía las capacidades del paradigma de programación orientada a agentes existente. Esta breve nota revisa primero los intentos ya realizados para definir la abstracción en IA y en sistemas de agentes. Para después, introducir nuestro modelo en forma de seis definiciones que cubren los conceptos de agentes, conocimiento anotado, utilidad y sociedad de agentes. La presentación es puramente cognoscitiva e intuitiva.

## 1. Introduction

Abstraction can be viewed as a methodology to express a system in terms of one selected theoretical framework. The word system covers software or hardware components of a computer system. In this process, the relevant features are identified and deliver the core of the abstraction while irrelevant ones are also identified and omitted. The theoretical framework is very often mathematics but this is not required. One may wish to abstract for instance what a society of agents is or whether a concept of emotion can be defined for agents. The concept of multi-agent has emerged as a paradigm for designing complex software systems. It is mainly used to better formalize problems in Distributed Artificial Intelligence (DAI) [19]. This leads us to center our analysis on both artificial intelligence and on agent systems, the emphasis being on the latter. An agent is very generally defined as (Wooldridge in [19]) "'. . . a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives"'. Then, various definitions have been proposed. For instance, in restricted situations of Game Theory one can adopt this definition [5]: "'An agent is an entity which can receive information about a state of the environment, take actions which may alter that state, and express preferences among the various possible states. The preferences are encoded for each agent by a utility function, a mapping from the set of all states to $\mathbb{R}$" "'. More simply, agents are able to act without any external control: they have their own control over their behavior and internal states in any possible environment. We adopt this latter approach.

Presentado por Luis M. Laita.
*Recibido*: February 5, 2004. *Aceptado*: Ocotber 13, 2004.
*Palabras clave / Keywords*: Abstraction, Agent, Annotated Knowledge, Ontology, Society of Agents, Utility Functions
*Mathematics Subject Classifications*: 68T01, 68T35, 68M14, 68Q60.

We propose a concept of Agent-Oriented Abstraction (AOA) that encompasses the features that are associated to the concept of agents and that is compliant with a societal approach of agents. The AOA concept couples two components in the definition of an agent. The first component is the knowledge part while the second is a decision mechanism. While there is no real agreement in the MAS community on the definition for an agent, it is in particular difficult to find a smooth transition from an agent to a society of agents. Shoham [18] proposed ten years ago a new computational framework, called Agent-Oriented Programming (AOP) which is introduced as a generalization of object-oriented programming (OOP). It seems that his main motivation was societal. AOA also provides an approach to the definition of a society of agents based on a classical theory in Sociology. The paper is structured as follows: We start by reviewing the relevant aspects of abstraction in Artificial Intelligence. This obviously includes works centered on agents. Then, we describe our model of Agent-Oriented Abstraction. We conclude with some illustrative applications that are under development.

## 2. Abstractions in Artificial Intelligence

It is a tautology to assume that a program is an abstract specification of a machine. Then, a computer is a decision machine where the decision is a possible resolution among potential actions. This point of view is expressed for instance in Wellman [20]. About twenty years ago, Newell [15] proposed that a central characteristic of practical AI is a particular abstraction level at which we interpret the behavior of intelligent machines. This led him to the concept of a basic rationality principle stating that an agent will select the action that leads to one of its goals. Wellman's thesis is that the rationality abstraction distinguishes economy among other social sciences. This leads him to point out the economic approach to AI that is extensively used in agent systems. In fact, rationality is a characteristic of game theory [16] where the expected utility functions, formalized in 1920 by von Neumann and Morgenstern [14], play a basic role when assessing possible strategies and are mandatory to define the concept of Nash's equilibrium [13]. Expected utility functions are illustrative examples of an abstraction mechanism for the concept of usefulness of actions performed by agents. One of the main resulting features is that a Nash equilibrium always exists for some models, and can be computed. A very recent presentation can be found in [5]. This is a prototypical example of what an abstraction can be in agent systems. A theory of abstraction in AI has been proposed by Giunchiglia and Walsh [9] in 1992. They mention that some synonyms of "'abstract"' are brief, synopsis and sketch while "'to abstract"' may mean to detach and also to separate. This leads to relate the process of abstraction to the process of separating, extracting from a representation another representation that consists of a brief sketch of the original representation. This is thus an intuitive or cognitive concept for an abstraction arising from common sense reasoning. It is translated in a very formal theory that they apply mainly to deduction or theorem proving. This approach announces the abstraction of OMRS (Open Mechanized Reasoning System) that was later generalized to symbolic computation under the name of OMCS (Open Mechanized Computation System) [1]. Although we will not comment further, AOA subsumes such an approach. The same authors published later [8] on a survey of theories of abstraction but also centered on deduction.

In the domain of multi-agent systems an abstraction for computational organizations is presented in [21]. These organizational abstractions are divided into organizational rules, organizational structures and organizational patterns. A main motivation is to deal with agents that are inherently cooperative towards one another. This is a refined evolution of a general trend in agent-oriented methodologies to view the structure of a multi-agent system in terms of a role model assigned to each agent. An agent is autonomous and plays a specific role. Several references relevant to this approach are given in [21].

Object-oriented methodology is a form of abstraction that is almost the default one nowadays. The following features can be found in any tutorial on OOP. An object is a bundle of variables and related methods. A method is a procedure that can modify the object behavior through its action on the variables. A prototype or blueprint for an object is a class that defines the variables and the methods common to all objects of the same kind. In fact, it is a programming language way to define a data type. The main benefits are encapsulation (the hiding of the code), inheritance (between parent and child classes) and polymorphism. We have thus a message passing system adapted to what we may call the environment through an interface. Wooldridge summarizes the relationship between the concepts of objects and agents in [19]. Shoham [18] proposed the concept of agent-oriented programming (AOP) to describe intelligent agents. He defines an agent as ”‘an entity whose state is viewed as consisting of mental components such as beliefs, capabilities, choices and commitments”’. In section 3**.** we will agree with a comment of McCarthy that mental attributes should be ascribed to objects when they are deemed useful. Furthermore, AOP is multi-threaded since agents can select on behaviour out of several possible ones while OOP is single-threaded since objects have no action choice. A simple analysis shows that this approach is societal: it is based on the interaction between an agent and its environment, for instance through the BDI (Belief, Desire, Intention) paradigm. Shoham assumes that an AOP system consists of three main components: a formal language to describe mental states, a programming language to define agents and a methodology to transform applications not involving actions into agents. He designs the AgentO language. Such remarks suggest that AOP subsumes OOP but is not a true abstraction mechanism. This is probably why agent platforms are mainly implemented in OOP languages. Contrary to OOP, AOP did not cause any breakthrough, although the languages PLACA of Thomas and AgentK [6] of Davies and Edwards have improved the message passing mechanism of AgentO. Nowadays, it is possible to find a number of papers dealing with an OO approach. A recent one is [10] where references can be found. An important remark is that AOP does not cover all possible communication mechanisms an agent may have with its environment whether they are pro-active, reactive, synchronous or not, BDI, speech acts, auctions, market simulations just to cite a few of them. None of the methodologies outlined here is suitable to design a concept of abstraction of agent methodologies.

## 3. The Agent-Oriented Abstraction Model

The previous section on abstraction through agent-oriented programming shows that the paradigm of abstraction is linked to a model for the relationship between agents and their environments. We partly agree with this statement but we propose a different abstraction methodology relying on a sociological model for a society. The presentation we give here is solely intuitive and is summarized by the following definitions. A theoretical presentation in terms of category theory ought to be possible but is a long term project.

**Definition 1** *An agent is an entity made of annotated knowledge coupled to a decision mechanism.*

**Definition 2** *The decision mechanism of an agent is the process by which an agent can reach its assigned goals. It is based upon the contents of the knowledge component. A decision mechanism is characterized by its utility.*

**Definition 3** *Knowledge annotations are classes or types structuring the knowledge possessed by or associated to agents.*

**Definition 4** *The utility of a decision mechanism is a measure of the efficiency of this mechanism. It is structured into utility classes.*

**Definition 5** *A society of agents is the societal organization arising from the actions performed by individual agents in the agent world assigned to a problem.*

**Definition 6** *A specialization is an implementation of the abstract classes for knowledge or utility.*

A first motivation for this abstraction mechanism is to have a continuous translation from a single agent system to a society of agents. Usual approaches for the concept of agent societies are based either on a too large number of agents or on a fixed description of the relationship between agents and their environment. We base our approach on Sociology and more precisely on the founding work of Weber stating that a society is the result of the actions of individuals. A more detailed description is given in [2]. A crucial point is that this approach implies the abstraction model sketched here. Our definition of agents subsumes the definitions that we have cited previously. An initial remark is that it may appear either familiar or even obvious to many experts of agent systems since associating an action performed by an agent to some sort of knowledge possessed by this agent is routinely found in the literature. We do not require agents to be autonomous since the requirement that agents possess a decision mechanism implies some autonomy that is only restricted by possible additional design requirements like enforced communication with the originator for security reasons. Indeed, the result of the decision is to fulfill the goals set to the agent, without any external contribution not assigned in the decision mechanism. Two extreme examples of agents are thermostats and mediator query systems. A thermostat has a knowledge base comprising a scale of temperatures while its decision mechanism is to put the heater on or off. Nowadays, intelligent agents are often modeled as mediator system querying information sources. The knowledge component is obvious while the decision mechanism is a query coupled to an inference algorithm. The KOMET system ([3], [11]) is an example of an intelligent agent.

The ”‘utility”’ of an action is an intuitive concept. When stating that a society is the result of aggregating the actions of the agents we imply such a concept. There is a scope of possible definitions of utility. Theoretically we may rely on the expected utility function as defined by von Neumann and Morgenstern. There is a new function with each new application. The trouble is that this is a problem in the theory of functions and to prove their existence is challenging. This makes their use fully impossible as a general tool. A way out is to express the actions as well-founded formulae of some logic and then to run a resolution algorithm. We are investigating this approach. Another method is model building as it is nicely and simply illustrated in [7]. To acknowledge such cases we introduce the concept of classes. A list of classes follows. It can be extended and is by no means exhaustive.

The concept of specialization is inspired by some methods used to design algorithms in algebraic geometry to find the Groebner bases of ideals and thus the solutions of polynomial equations. It amounts there to set values to parameters.

1. The expected utility functions. This is the class of functions defined in [14]. The specialization is thus a function. It is however usually very difficult to prove the existence of such functions.

2. The common sense measures of usefulness. Specialization ranges from fuzzy logic to probabilistic expectation through belief assessment or any market mechanism.

3. The class of models. For instance, in [7] a model inspired by Physics is proposed. Specialization is then a function arising from a model. A difference with expected utility functions is that these functions always do exist when the model exists.

4. The class arising from logical modeling and running a resolution algorithm on formulae of a logic. A very simple model under investigation is first order logic modeling in game theory.

These classes imply a hierarchical organization of the concept of utility. The root is ”‘utility”’ and the sons are the classes listed here. Then, a specialization is inherited from a class. This means that the sons of a class are segmented into diverse specializations.

Classical models for e-commerce do associate an ontology to any agent. An ontology is a structured knowledge base. While it is usually possible to find out the taxonomy part of an ontology, it is not so easy to identify the proper structures of this ontology. A theoretical approach coming from the concept of distance and based on the entropy for probabilistic states is proposed in [4]. Even this approach is not general enough to state that the knowledge content of an agent is solely an ontology. If this knowledge has most of the time an ontological component, it also includes information on the communication mechanisms between an agent and its environment. An advantage of defining societies on purely sociological ground is that communication and negotiation protocols can be seen as part of the knowledge of an agent. Typing knowledge has been introduced several years ago in the design of KOMET (but never described in a publication). It allows to deal with structured and unstructured knowledge in a same setting. We introduce the concept of annotated knowledge to structure the knowledge component of an agent. An annotation is a class, also called a type, of knowledge. Annotations enable a hierarchical partitioning of knowledge. It ought to be obvious that the choice of "annotated" is motivated by the analogy with generalized annotated programming that has been adopted to design KOMET.

The annotations for knowledge are divided into the following main classes. As for the concept of utility the following list is by no means exhaustive. It can be extended. While in object-oriented programming it is meaningful to predefine objects and classes, in artificial intelligence we cannot predefine all existing agents (objects) and bound the number of their annotations and associated classes.

1. Ontology. This annotation covers ontologies and knowledge bases. It is divided into classes corresponding to structured, unstructured, complete or incomplete knowledge/ontology. Specialization of such classes is obtained through a query, information retrieval or data mining system. A specialization of these classes may arise frm the methodology outlined in [17].

2. Communication. It is expressed through classes that describe the main facets of the communication mechanisms between agents and with the external world if any. Coordination is achieved through subclasses for cooperation, planning (both distributed and centralized), competition or negotiation. The specialization of these classes is performed when implementing protocols, for instance interaction protocols or task distribution protocols. These protocols can be synchronous or asynchronous, they can be binary or n-ary protocols. The number of possible protocols is huge. It is even possible to design a communication protocol by means of an agent system connecting the communicative entities within an agent system. Message types are also among the specializations found in communication. A specialization for negotiation can be based upon the market mechanism, a contract net approach or a blackboard system.

3. Cognition. There is obviously a cognitive component among the possible annotations of knowledge. We distinguish three main classes: semantics, pragmatics and models. Semantics implies a formalized representation of the cognition found in an agent. This translate into specializations that could be logic-based or a well defined view maintenance mechanism. The class "‘pragmatics"’ relates to common sense reasoning and reasoning which is not based on a sound theoretical model. The class "‘models"’ covers any cognitive meaning that although well captured in models is by no way theoretically certain. This leads to specializations as for example in terms of speech act where although the separation into locution, illocution and perlocution is well established, there is no proof it is a theoretically sound simulation of human communication. It is at best a meaningful approximation. This annotation covers the implementation features appearing under headlines such as: descriptive, prescriptive, reactive, pro-active, personal, conventional, objective or subjective, speaker versus hearer perspective. The BDI model (Belief, Desire and Intention) is set within this annotation.

4. Safety. This annotation could also be labeled integrity. It recognizes that any system is set into a real world where laws and rules governed the action of agents. A first class is obviously security. Regardless if for mobile or static agents, security issues are becoming overwhelmingly important. The specialization of this class is achieved for instance through the classical encryption methods or fire-wall technologies. A second class concerns laws and regulations. Whether within a country or a company, any representative agent is required to obey laws and regulations. Specializations are obvious through the (too many) existing regulations, directives and laws.

An additional comment on the latter annotation is that since abstractions in agent systems are linked to a concept of society, or at least to societal features, it is not possible to have a concept of society that is not self-protecting through security issues.

A main feature of the splitting of knowledge into annotations and then classes is that inheritance from different annotations and classes is possible. Indeed, the BDI approach is such an example. It is a cognitive annotation but also affects the communication annotation. We have thus inheritance from different classes possibly from separate annotations. This means that polymorphic typing is apparently possible. One must remain however very cautious with such a statement since to prove it is usually not straightforward. The message passing mechanism found in OOP is embedded into the communication annotation with some inherited properties from the safety and cognition annotations. For instance, we may want to allow messages in a negotiation protocol based upon belief maintenance and in agreement with the regulations governing the relevant society for this agent.

One of the main characteristics of a multiagent system is the architecture of this system. Classically, one considers logic-based architectures, reactive architectures, BDI architectures for practical reasoning systems and two kinds of layered architectures (horizontal and vertical). In addition, the simulation of animal societies leads to define architectures characterizing a given society. The best known example is probably the case of ants. In our approach, the architectures are consequences of the abstraction based upon the six definitions given previously. In particular, the knowledge annotations and their specializations determine almost directly the relevant architecture.

A last comment on the proposed approach is that it is fully generic. The simple inheritance mechanism we propose leads directly to such a generic flavor.

## 4. Conclusion

We have sketched a model for agent-oriented abstraction that generalizes the basic concepts present in object-oriented programming. It is general enough to cover the many facets of AI programming. Agents are seen as objects and through their knowledge contents they are organized into annotations that gather classes. Encapsulation, inheritance and polymorphism are features that can be adequately defined. Message passing is achieved through communication mechanisms that are included in the knowledge owned by an agent. The analogy of the methods found in objects of OOP is a decision mechanism that is by definition part of an agent. To validate this model we are investigating its application to the modeling of corporate knowledge [12]. This application provides a validation of the cognitive meaning of the concept of agent-oriented abstraction. It provides also a blueprint of a possible formulation of a specialized AOA in the language of category theory. In this direction, the investigation of the formal specification of an AOA model with universal algebra appears as a challenging but fascinating project. Indeed, this would provide a feasible approach to defining a formal semantic for a model one could easily extend to a semantic web model.

A longer report is in preparation. It will provide a better comparison of the concepts of OOP and AOA. Also, we will address some issues that have been purposely omitted here. In particular, the role of ”‘time”’ either through temporal logic or synchronization and the notions of single-threaded and multi-threaded objects and abstractions will be covered.

**Acknowledgement.** The first author thanks John Campbell and Paolo Petta for very meaningful comments on the first draft of this paper.

Work partially supported by the Calculemus Research Training Network, HPRN-CT-2000-00102. Région Rhône-Alpes, France, partially supported Pierre Maret.

## References

[1] P. G. Bertoli, J. Calmet, K. Homann and F. Giunchiglia. 1998. Specification and integration of theorem provers and computer algebra systems. *Fundamenta Informaticae* **39**:1-2, pp. 39–57.

[2] J. Calmet, A. Daemi, R. Endsuleit and T. Mie. 2004. A liberal approach to openess in societies of agents. *proc. of ESAW03, Engineering Societies in the Agents World,* LNAI **3071**, pp. 81–92.

[3] J. Calmet, S. Jekutsch, P. Kullmann and J. Schü. 1997. KOMET - a system for the integration of heterogeneous information sources. *International Symposium on Methodologies for Intelligent Systems*, LNAI **1325**, pp. 318–327.

[4] J. Calmet and A. Daemi. 2004. From entropy to ontology. *Fourth International Symposium "'From Agent Theory to Agent Implementation"'* **2**, pp. 547-551.

[5] R. S. Datta. 2003. Algebraic methods in game theory. PhD thesis. UCLA at Berkeley.

[6] W.H.E. Davies and P. Edwards. 1994. Agent-K: An integration of AOP and KQML. *Proc. of the CIKM'94 Workshop on Intelligent Agents*.

[7] M. Detyniecki, B. Bouchon-Meunier and R. R. Yager. 1999. Balance operator: a new vision on aggregation operators. *Proc. of the Joint EUROFUSE-SIC International Conference*, pp. 242–246.

[8] F. Giunchiglia, A. Villaforita and T. Walsh. 1997. Theories of abstraction. *Technical report* **97-0051**, DIST Trento.

[9] F. Giunchiglia and T. Walsh. A theory of abstraction. *Artificial Intelligence* **57**:2-3, pp. 323–390.

[10] B. Henderson-Sellers, P. Giorgini and P. Bresciani. 2004. Enhancing agent OPEN with concepts used in the tropos methodology. *Proc. of ESAW03, Engineering Societies in the Agents World*, LNAI **3071**, pp. 328 – 345.

[11] P. Kullmann. 2001. Wissensrepräsentation und Anfragebearbeitung in einer logikbasierten Mediatorumgebung. PhD thesis. Univ. of Karlsruhe.

[12] P. Maret, M. Hammond and J. Calmet. 2004. Agent Societies for Corporate Knowledge Issues. To be presented at the *Fifth International Workshop on Engineering Societies in the Agents World (ESAW)*.

[13] J. Nash. 1950. Equilibrium points in n-person games. *Proc. of the Nat. Academy of Science of the USA* **36**, pp. 48–49.

[14] J. von Neumann and O. Morgenstern. 1980. *Theory of games and economic behavior*, Princeton University Press.

[15] A. Newell. 1982. The knowledge level. *Artificial Intelligence* **18**, pp. 87–127.

[16] M. J. Osborne and A. Rubinstein. 1994. *A course in games theory*, MIT Press.

[17] J.M. Serrano, S. Ossowski and A. Fernandez. 2003. The pragmatics of software agents: Analysis and design of agent communication languages. *Intelligent Information Agents: The AgentLink Perspective*, Springer Verlag, pp. 234–273.

[18] Y. Shoham. 1993. Agent-oriented programming. *Artificial Intelligence* **60**, pp. 51–92.

[19] G. Weiss. 1999. *Multiagent systems: a modern introduction to distributed artificial intelligence*, MIT Press.

[20] M. P. Wellman. 1995. The economic approach to artificial intelligence. *ACM Comp. Surveys* **27**:3, pp. 360–362.

[21] F. Zambonelli, N. J. Jenninngs and M. Wooldridge. 2001. Organisational rules as an abstraction for the analysis and design of multi-agent systems. *International Journal of Software Engineering and Knowledge Engineering* **11**:3, pp. 303–328.

Jacques Calmet; Regine Endsuleit
IAKS
University of Karlsruhe (TH)
Am Fasanengarten 5
D-76 128 Karlsruhe, Germany
{calmet,endsuleit}@ira.uka.de

Pierre Maret
LIRIS CNRS/FRE 2672
INSA de Lyon
7 avenue Jean Capelle
F-69621, Villeurbanne, France
pierre.maret@liris.cnrs.fr