Agent Oriented Programming

an overview of the framework and summary of recent research

Yoav Shoham
Robotics Laboratory
Computer Science Department
Stanford University

This is a short overview of the *agent-oriented programming* (AOP) framework. AOP can be viewed as an specialization of *object-oriented programming*. The state of an agent consists of components called beliefs, choices, capabilities, commitments, and possibly others; for this reason the state of an agent is called its *mental state*. The mental state of agents is captured formally in an extension of standard epistemic logics: beside temporalizing the knowledge and belief operators, AOP introduces operators for commitment, choice and capability. Agents are controlled by *agent programs*, which include primitives for communicating with other agents. In the spirit of *speech-act theory*, each communication primitives is of a certain type: informing, requesting, offering, and so on. This document describes these features in a little more detail, and summarizes recent results and ongoing AOP-related work.

1 Introduction

Agent Oriented Programming is a proposed new programming paradigm, based on a societal view of computation. Although new, the proposal benefits from extensive previous research. Indeed, the discussion here touches on issues that are the subject of much current research in AI, issues which include the notion of agenthood and the relation between a machine and

its environment. Many of the ideas here intersect and interact with the ideas of others. In this overview, however, I will not place this work in the context of other work. That, as well as more details on AOP, appear in a Technical Report STAN-CS-1335-90 (revised), and subsequent publications which are mentioned below.

1.1 What is an agent?

The term 'agent' is used frequently these days. This is true in AI, but also outside it, for example in connection with data bases and manufacturing automation. Although increasingly popular, the term has been used in such diverse ways that it has become meaningless without reference to a particular notion of agenthood. Some notions are primarily intuitive, others quite formal. In several longer publications I outline several senses of agenthood that I have discerned in the AI literature. Given the limited space, here I will directly present "my" sense of agenthood.

I will use the term '(artificial) agents' to denote entities possessing formal versions of mental state, and in particular formal versions of beliefs, capabilities, choices, commitments, and possibly a few other mentalistic-sounding qualities. What will make any hardware or software component an agent is precisely the fact that one has chosen to analyze and control it in these mental terms.

The question of what an agent is now replaced by the question of what can be described in terms of knowledge, belief, commitment, *et cetera.* The answer is that *anything* can be so described, although it is not always advantageous to do so. D. Dennett proposes the "intentional stance," from which systems are ascribed mental qualities such as intentions and free will. The issue, according to Dennett, is not whether a system really is intentional, but whether we can coherently view it as such. Similar sentiments are expressed by J. McCarthy in his 'Ascribing Mental Qualities to Machines' paper, who also distinguishes between the 'legitimacy' of ascribing mental qualities to machines and its 'usefulness.' In other publications I illustrate the point through the light-switch example. It is perfectly coherent to treat a light switch as a (very cooperative) agent with the capability of transmitting current at will, who invariably transmits current when it believes that we want it transmitted and not otherwise; flicking the switch is simply our way of communicating our desires. However, while this is a coherent view, it does not buy us anything, since we essentially understand

the mechanism sufficiently to have a simpler, mechanistic description of its behavior. In contrast, we do not have equally good knowledge of the operation of complex systems such as robots, people, and, arguably, operating systems. In these cases it is often most convenient to employ mental terminology; the application of the concept of 'knowledge' to distributed computation, discussed below, is an example of this convenience.

1.2 Agent- versus Object-Oriented Programming

Adopting the sense of agenthood just described, I have proposed a computational framework called *agent-oriented programming* (AOP). The name is not accidental, since from the engineering point of view AOP can be viewed as a specialization of the *object-oriented programming* (OOP) paradigm. I mean the latter in the spirit of Hewitt's original Actors formalism, rather than in some of the senses in which it used today. Intuitively, whereas OOP proposes viewing a computational system as made up of modules that are able to communicate with one another and that have individual ways of handling incoming messages, AOP specializes the framework by fixing the state (now called *mental state*) of the modules (now called *agents*) to consist of precisely-defined components called beliefs (including beliefs about the world, about themselves, and about one another), capabilities, choices, and possibly other similar notions. A computation consists of these agents informing, requesting, offering, accepting, rejecting, competing, and assisting one another. This idea is borrowed directly from the *speech act* literature. Speech-act theory categorizes speech, distinguishing between informing, requesting, offering and so on; each such type of communicative act involves different presuppositions and has different effects. Speech-act theory has been applied in AI, in natural language research as well as in plan recognition. To my knowledge, AOP and McCarthy's Elephant2000 language are the first attempts to base a programming language in part on speech acts. Figure 1 summarizes the relation between AOP and OOP.[1]

[1]There is one more dimension to the comparison, which I omitted from the table, and it regards inheritance. Inheritance among objects is today one of the main features of OOP, constituting an attractive abstraction mechanism. I have not discussed it since it is not essential to the idea of OOP, and even less so to the idea of AOP. Nevertheless a parallel can be drawn here too.

Framework:	OOP	AOP
Basic unit:	object	agent
Parameters defining state of basic unit:	unconstrained	beliefs, commitments, capabilities, choices, ...
Process of computation:	message passing and response methods	message passing and response methods
Types of message:	unconstrained	inform, request, offer, promise, decline, ...
Constraints on methods:	none	honesty, consistency, ...

Figure 1: OOP versus AOP

1.3 On the use of pseudo-mental terminology

The previous discussion referred to mentalistic notions such as belief and commitment. In order to understand the sense in which I intend these, it is instructive to consider the use of logics of knowledge and belief in AI and distributed computation. These logics, which were imported directly from analytic philosophy first to AI and then to other areas of computer science, describe the behavior of machines in terms of notions such as knowledge and belief. In computer science these mentalistic-sounding notions are actually given precise computational meanings, and are used not only to prove properties of distributed systems, but to program them as well. A typical rule in such a 'knowledge based' systems is "if processor A does not *know* that processor B has received its message, then processor A will not send the next message." AOP augments these logics with formal notions of choices, capabilities, commitments, and possibly others. A typical rule in the resulting systems will be "if agent A *believes* that agent B has *chosen* to do something harmful to agent A, then A will *request* that B change its choice." In addition, temporal information is included to anchor belief, choices and so on in particular points in time.

Here again we may benefit from some ideas in philosophy and linguistics. As in the case of knowledge, there exists work in exact philosophy on logics for choice and ability. Although they have not yet had an effect in AI comparable to that of logics of knowledge and belief, they may in the future.

Intentional terms such as knowledge and belief are used in a curious sense in the formal AI community. On the one hand, the definitions come nowhere close to capturing the full

linguistic meanings. On the other hand, the intuitions about these formal notions do indeed derive from the everyday, common sense meaning of the words. What is curious is that, despite the disparity, the everyday intuition has proven a good guide to employing the formal notions in some circumscribed applications. AOP aims to strike a similar balance between computational utility and common sense.

2 Overview of the AOP framework

A complete AOP system will include three primary components:

- A restricted formal language with clear syntax and semantics for describing mental state. The mental state will be defined uniquely by several modalities, such as belief and commitment.
- An interpreted programming language in which to program agents, with primitive commands such as `REQUEST` and `INFORM`. The semantics of the programming language will depend in part on the semantics of mental state.
- An 'agentifier,' converting neutral devices into programmable agents.

In the remainder of this document I will start with an short discussion of mental state. I will then present a general family of agent interpreters, a simple representative of which has already been implemented. I will end with a summary of recent research results and outstanding questions related to AOP.

3 On the mental state of agents

The first step in the enterprise is to define agents, that is, to define the various components of mental state and the interactions between them. There is not a unique 'correct' definition, and different applications can be expected to call for specific mental properties.[2]

[2]In this respect our motivation here deviates from that of philosophers. However, I believe there exist sufficient similarities to make the connection between AI and philosophy mutually beneficial.

In related past research by others in AI three modalities were explored: belief, desire and intention (giving rise to the pun on BDI agent architectures). Other similar notions, such as goals and plans, were also pressed into service. These are clearly important notions; they are also complex ones, however, and not necessary the most primitive ones. Cohen and Levesque, for example, propose to reduce the notion of *intention* to those of *goal* and *persistence*. We too start with quite basic building blocks, in fact much more basic that those mentioned so far. We currently incorporate two modalities in the mental state of agents: *belief* and *obligation* (or *commitment*). We also define *decision* (or *choice*) as an obligation to oneself. Finally, we include a third cateogry which is not a mental construct *per se*, *capability*. There is much to say on the formal definitions of these concepts; some of results described in the final section address this issue.

By restricting the components of mental state to these modalities I have in some informal sense excluded representation of motivation. Indeed, we do not assume that agents are 'rational' beyond assuming that their beliefs, obligations and capabilities are internally and mutually consistent. This stands in contrast to the other work mentioned above, which makes further assumptions about agents acting in their own best interests, and so on. Such stronger notions of rationality are obviously important, and I am convinced that in the future we will wish to add them. However, neither the concept of agenthood nor the utility of agent-oriented programming depend on them.

4 A generic agent interpreter

In the previous section I discussed the first component of the AOP framework, namely the definition of agents. I now turn to the central topic of this paper, the programming of agents, and will outline a generic agent interpreter.

The behavior of agents is governed by programs; each agent is controlled by his own, private program. Agent programs themselves are not logical entities, but their control and data structures refer to the mental state of the agent using the logical language.[3]

[3]However, an early design of agent programs by J. Akahani was entirely in the style of logic programming; in that framework program statements themselves were indeed logical sentences.

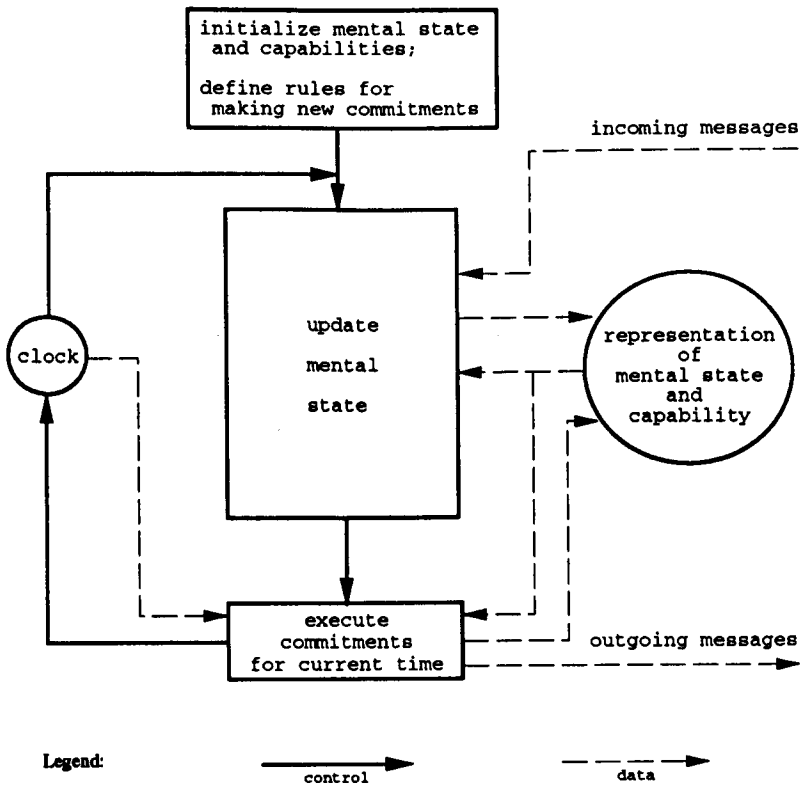

Figure 2: A flow diagram of a generic agent interpreter

The basic loop

The behavior of agents is, in principle, quite simple. Each agent iterates the following two steps at regular intervals:

1. Read the current messages, and update your mental state (including your beliefs and commitments);

2. Execute the commitments for the current time, possibly resulting in further belief change. Actions to which agents are committed include communicative ones such as informing and requesting.

The process is illustrated in Figure 2; dashed arrows represent flow of data, solid arrows temporal sequencing.

5 Summary of results and ongoing research

A more detailed discussion of AOP appears in [7]; the implemented interpreter is documented in [13]. Ongoing collaboration with the Hewlett Packard corporation is aimed at incorporating features of AOP in the New Wave™ architecture.

Preliminary ideas on the logic of mental state appear in [12]; a concrete proposal is made in [11]. This latter work addresses the properties of mental state at a given moment. Other publications address dynamic aspects of mental state. A logic for perfect memory and justified learning is discussed in [5]. [1] addresses the logic of belief revision; specifically, the postulates of belief update are shown to be derivable from a formal theory of action. The theory used there is the 'provably correct' theory presented in [3], which was later generalized to a frameowrk admitting concurrent action [4].

In parallel to the logical aspects of action and mental state, we have investigated algorithmic questions. We have proposed a specific mechanism for tracking how beliefs change over time, called *temporal belief maps* [2].

We are particularly interested in how multiple agents can function usefully in the presence of other agents. In [6] we propose the mechanism of *protograms* to balance conflicting influences of different agents. We are also interested in minimizing such conflicts in the first place, and have been investigating the computational utility of social law. In [10] we study the special case of traffic laws in a restricted robot environment; in [8] we propose a general framework for representing social laws within a theory of action, and investigate the computational complexity of automatically synthesizing useful social laws.

References

[1] A. del Val and Y. Shoham. Deriving the postulates of belief update from theories of action. Stanford working document, 1992.

[2] H. Isozaki and Y. Shoham. A mechanism for reasoning about time and belief. In *Proc. Conference on Fifth Generation Computer Systems*, Japan, 1992.

[3] F. Lin and Y. Shoham. Provably correct theories of action (preliminary report). In *Proc. NCAI*, Anaheim, CA, 1991.

[4] F. Lin and Y. Shoham. Concurrent actions in the situation calculus. In *Proc. NCAI*, San Jose, CA, 1992.

[5] F. Lin and Y. Shoham. On the persistence of knowledge and ignorance. Stanford working document, 1992.

[6] E. Mozes and Y. Shoham. Protograms. Stanford working document, 1991.

[7] Y. Shoham. Agent Oriented Programming. *The Artificial Intelligence Journal*, 1992 (to appear).

[8] Y. Shoham and M. Tennenholtz. On the synthesis of useful social laws for artificial agents. In *Proc. NCAI*, San Jose, CA, 1992.

[9] Y. Moses and Y. Shoham. Belief as Defeasible Knowledge. *The Artificial Intelligence Journal*, 1992 (to appear; preliminary version appeared in IJCAI 89).

[10] Y. Shoham and M. Tennenholtz. On traffic laws for mobile robots. Stanford working document, 1992.

[11] B. Thomas. A logic for representing action, belief, capability, and intention, 1992. Stanford working document.

[12] B. Thomas, Y. Shoham, A. Schwartz, and S. Kraus. Preliminary thoughts on an agent description language. *International Journal of Intelligent Systems*, 6(5):497–508, August 1991.

[13] M. Torrance. The AGENT0 programming manual, 1991. Stanford technical report.

[14] Y. Shoham and M. Tennenholtz. Emergent Conventions in Multi-Agent Systems. In *Proc. KR*, Boston, MA, 1992.