CIFE CENTER FOR INTEGRATED FACILITY ENGINEERING

Agent-Oriented Programming

by

Yoav Shoham

TECHNICAL REPORT
Number 42

November, 1990

Stanford University

If you would like to contact the authors please write to:

c/o CIFE, Civil Engineering,
Stanford University,
Terman Engineering Center
Mail Code: 4020
Stanford, CA 94305-4020

Agent-Oriented Programming

Yoav Shoham
Robotics Laboratory
Computer Science Department
Stanford University

A new computational framework is presented, called *agent-oriented programming*, which can be viewed as an specialization of *object-oriented programming*.[1] 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 presents the concept of AOP, and describes progress made to date towards realizing it.

ERRATA in

Agent-Oriented Programming

The report contains several typographical errors. Most of them are innocuous, but a couple are not. Here are the more misleading ones that have been discovered to date (thanks to Fangzhen Lin).

On page 27, please replace the pattern `(B,employee(smith,acme))` by the pattern `(B,[t,employee(smith,acme)])`.

Later in that same page, replace the line

```
IF (B,employee(smith,acme)) THEN INFORM(t,a,employee(smith,acme))
```

by

```
IF (B,[t',employee(smith,acme)]) THEN INFORM(t,a,[t',employee(smith,acme)])
```

On page 38, the first macro contains two occurrences of the term `issue_bp`; please replace the second occurrence by `physical_issue_bp`.

At the bottom of the same page, replace < by >.

Finally, in the first commitment rule on page 39, replace the second argument `((B,?p))` by the argument `true`.

Contents

1 Introduction

This manuscript proposes a new programming paradigm, based on a societal view of computation. I will describe the concept as well as the progress I, together with others in the Agents group within the Robotics Laboratory, have made to date towards realizing this concept. The discussion touches on issues that are the subject of much current research in AI, issues that 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. For the sake of continuity, however, I will not place this work in the context of other work until the end.

1.1 What is an agent?

The term 'agents' 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. Some are very austere, defining an agent in automata-theoretic terms, and others use a more lavish vocabulary. Below are three overlapping senses of agenthood that I have discerned in the AI literature.

- Agents are subservient. This is perhaps the original sense of the word, someone acting on behalf of someone else. This is the sense of 'agency theory' in economics, where considerations include calculations of incentives and so on (of courses, agency theory in economics encompasses much more than just this sense). Early on this was the sense in AI too, but few of the people in AI today who use the word intend this meaning. (One exception that comes to mind is the use of the word by people in the intelligent-interfaces community, when they talk of 'software agents' carrying out the user's wishes.)

- Agents function continuously and 'autonomously' in an environment in which processes take place and other agents exist. This is perhaps the only property that is assumed uniformly by those in AI who use the term. The sense of 'autonomy' is not precise,

but the term is taken to mean that the agents' activities do not require constant human guidance or intervention. Often certain further assumptions are made about the environment, for example that it is physical and partially unpredictable. In fact, agents are sometimes taken to be robotic agents, in which case other issues such as sensory input, motor control and time pressure are mentioned.

- Agents are "high-level." Although this sense is quite vague, many take some version of it to distinguish agents from other software or hardware components. The "high level" is manifested in symbolic representation and/or some cognitive-like function: agents may be 'informable' (Genesereth [20]), may contain symbolic plans in addition to stimulus-reponse rules (Dean [11], Hayes-Roth et al. [25], Drummond [14], Mitchell [38], and many others), may even possess natural-language capabilities, and so on. This sense is *not* assumed uniformly in AI, and in fact a certain counter-ideology deliberately denies any representation in agents. This camp tends be included in the camp most inclined to view agents as real-time robotic agents (some representatives: Brooks [6], Schoppers [47], and Agre and Chapman's celebrated [1], roughly in decreasing order of fanaticism).

I take it that agents may, but need not, be subservient, and that, alongside other agents, they function in an environment that may, but need not, be partially unknown. I will not assume anything further about the environment; in particular I will not assume that the environment is physical nor that agents are under time pressure (although as a special case I am interested in real-time robotic agents). Most crucially, I will use the term '(artificial) agents' to denote entities possessing formal versions of of mental state, and in particular formal versions of knowledge, 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; in this respect "my" agents too are high level.

The question of what an agent is now becomes the question of what can be described in terms of knowledge, belief, commitment, etcetera. The answer is that *anything* can be so described, although it is not always advantageous to do so. In [12] and other publications 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 [35], who also distinguishes between the 'legitimacy' of ascribing mental qualities to machines and its 'usefulness':

> To ascribe certain *beliefs, free will, intentions, consciousness, abilities* or *wants* to a machine or computer program is legitimate when such an ascription expresses the same information about the machine that it expresses about a person. It is useful when the ascription helps us understand the structure of the machine, its past or future behavior, or how to repair or improve it. It is perhaps never logically required even for humans, but expressing reasonably briefly what is actually known about the state of the machine in a particular situation may require menta l qualities or qualities isomorphic to them. Theories of belief, knowledge and wanting can be constructed for machines in a simpler setting than for humans, an d later applied to humans. Ascription of mental qualities is most straightforward for machines of known structure such as thermostats and computer operating systems, but is most useful when applied to entities whose structure is very incompletely known.

In [49] 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 robots, people, and, arguably, operating systems. In these cases it 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.[2]

[2]In [49] I discuss how the gradual elimination of animistic explanations with the increase in knowledge is correlated very nicely with both developmental and evolutionary phenomena. In the evolution of science, theological notions were replaced over the centuries with mathematical ones. Similarly, in Piaget's stages of child development, there is a clear transition from animistic stages around the ages of 4-6 (when, for example, children claim that clouds move because they follow us around), and the more mature later stages.

Framework:	OOP	AOP
Basic unit:	object	agent
Parameters defining state of basic unit:	unconstrained	knowledge, 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

1.2 Agent- versus Object-Oriented Programming

Adopting the sense of agenthood just described, I will propose 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 an specialization of the *object-oriented programming* (OOP) paradigm. I mean the latter in the spirit of Hewitt's original Actors formalism [26], rather than in the more specific sense 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 in-coming messages, AOP specializes the framework by allowing the modules, now called agents, to possess knowledge and beliefs about themselves and about one another, to have certain capabilities and make choices, and possibly other similar notions. A computation consists of these agents informing, requesting, offering, accepting, rejecting, competing with and assisting one another. Figure 1 summarizes the relation between AOP and OOP.[3]

1.3 AOP and logics of knowledge

The previous discussion offered a programming-paradigm perspective on AOP. An alternative view of AOP is as an application of a formal language. The formal language being applied is a generalization of epistemic logics, which have been used extensively in AI (cf. [39, 29, 32, 50])

[3]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, and I discuss it briefly in the final section.

and distributed computation (cf. [23, 24]). 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 as '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 *knows* 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 knowledge, choices and so on in particular points in time.

Here again we 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 (cf. [15]). More centrally, however, we borrow ideas from the *speech act* literature [4, 48, 22]. 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 [43, 28]. In a sense, AOP too can be viewed as a rigorous implementation of a fragment of direct-speech-act theory.

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.

It should be understood that this document describes a concept rather than a finished product. I believe that the concept is sound and that it can it can form a basis for future computer environments. I am also aware that some of the issues involved in implementing it are nontrivial and must be settled before a practical system is built. I will describe work

we are carrying out that addresses these issues.

Organization of the document

The rest of the document is organized as follows. In Section 2 I provide further motivation for the AOP paradigm by looking at three futuristic applications. In Section 3 I outline the main ingredients of the AOP framework. The bulk of the paper then describes progress we have made to date towards realizing the concept. In Section 4 I discuss the formal definition of mental state. In Section 5 I present a simple language called AGENT0 in which to program agents. In Section 6 I discuss the compilation of agent programs into a neutral process language. In Section 7 I briefly discuss related work by others in AI. Finally, in the last section I discuss some of the directions in which the work described here can be extended.

2 Three scenarios

Below are three semi-futuristic scenarios. Although they are couched in different settings, the three all illustrate similar points. They each involve communicative acts such as informing, requesting, committing, permitting and commanding, and require agents to reason about the beliefs, capabilities and commitments other agents.

2.1 Manufacturing automation

Alfred and Brenda work at a car-manufacturing plant. Alfred handles regular-order cars, and Brends handles special-order ones. The plant has a welding robot, Calvin. The overly busy plant foreman has written a coordinating program, Dashiel. The following scenario develops, involving communication between Alfred, Brenda, Calvin and Dashiel.

8:00: Alfred requests that Calvin promise to weld ten bodies for him that day; Calvin agrees to do so.

8:30: Alfred requests that Calvin accept the first body, Calvin agrees, and the first body arrives. Calvin starts welding it and promises Alfred to notify him when it is ready for the next body.

8:45: Brenda requests that Calvin work on a special-order car which is needed urgently. Calvin reponds that it cannot right then, but that it will when it finishes the current job, at approximately 9:00.

9:05: Calvin completes welding the Alfred's first car, ships it out, and offers to weld Brenda's car. Brenda ships it the car, and Calvin starts welding.

9:15: Alfred enquires why Calvin is not yet ready for his (Alfred's) next car. Calvin explains why, and also that it (Calvin) expects to be ready by about 10:00.

9:55: Calvin completes welding Brenda's car, and ships it out. Brenda requests that it reaccept it and do some painting, but Calvin refuses, explaining that it does not know how to paint. Calvin then offers to weld another car for Alfred, and proceeds to weld Alfred's cars for a while.

12:15: Brenda requests that Calvin commit to welding four more special-order cars that day. Calvin replies that it cannot, since that conflicts with its commitment to Alfred, who still has six unwelded cars. Brenda requests Alfred to release Calvin from its commitment to Alfred. Alfred refuses. Brenda requests that Dashiel (remember Dashiel?) order Calvin to accept her important request and revoke its commitment to Alfred. Dashiel orders Calvin to weld two of Brenda's cars, and then as many of Alfred's as time allows.

2.2 Robot coordination

It is 1999, and the new Information Sciences building at Stanford has been completed. In addition to its human inhabitants, the building is populated by about 100 gofer robots. The role of the gofers is to carry documents to and from the copying machine, fetch coffee and sodas, and generally make themselves useful.[4]

At the junction of two corridors two gofers, $Gofer_1$ and $Gofer_2$, engage in the following exchange.

$Gofer_1$: I intend to turn into the north corridor.

[4] "Gofer" is an actual ongoing project in the robotics laboratory, under the direction of J. C. Latombe. The completion of the building by the said date, however, is pure speculation.

$Gofer_2$: So do I; you may go first. Where are you headed?

$Gofer_1$: The copying machine.

$Gofer_2$: Will you xerox this document for me and drop it off at John's office?

$Gofer_1$: Okay.

$Gofer_1$ proceeds down the north corridor, and $Gofer_2$ scuttles back down the west corridor. Halfway down the corridor it runs into $Gofer_3$.

$Gofer_3$: Where are you headed?

$Gofer_2$: To John's office.

$Gofer_3$: Watch out for a traffic jam in corridor 25.

2.3 Airline reservation

This example is a minor modification of one due to John McCarthy [36], who uses it to illustrate his Elephant programming language (see discussion of Elephant in Section 7). The example involves three agents: a passanger P, an airline clerk C, and C's supervisor S. P and S are human, and C is a program. Or is it the other way around. Or perhaps all three are programs.

March

P to C: Please inform me what flights you have from San Francisco to New York on April 18.

C to P: Flight #354 departs at 08:30, flight #293 departs at 10:00, flight #441 departs at noon, ...

P to C: Please book me on #354.

C to P: That is sold out.

P to C: Please book me on #293.

C to P: That is confirmed; your reservation number is 112358.

P to C: Please book me also on #441.

C to P: That conflicts with #293; I am not allowed to double book a passanger.

P to C: Please get permission to do so.

C to S: I request permission for the following double booking: ...

S to C: Permission denied.

C to P: Sorry, I cannot get approval.

April 18, at the airport

P to C: My name is John Smith; I have a reservation for flight #293.

C to P: Here is your boarding pass.

3 Overview of the AOP framework

The preceding scenarios made reference to mentalistic notions such as knowledge, beliefs and commitments. Our goal now is to make engineering sense out of these abstract concepts. The result will be the agent-oriented programming paradigm. The AOP framework 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 derive from the semantics of mental state.
- A compiler from the agent-level language to an abstract model of processes.

In the remainder of the document I expand on these components, and describe progress made to date towards realizing them.

4 The definition of mental state[5]

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. In this section I will discuss what could be viewed as a bare-bones theory of mental state, a kernel that will in the future be modified and augmented. The work summarized here has been carried out jointly among several members of our research group, and reported initially in [51].

In related past research by others (see Section 7), three modalities were explored: belief, desire and intention (giving rise to the pun on BDI agent architectures). Although we started out by exploring similar notions, with time we lowered our sights. In this document I will concentrate on two primitive modalities – belief and commitment – and two derived ones – choice and capability – which we have found more basic (strictly speaking, capability is a relation between the mental state and the environment; more on that later).

By restricting the components of mental state to these modalities I have in some informal sense excluded representation of motivation. Indeed, I will not assume that agents are 'rational' beyond assuming that their beliefs, commitments and choices are internally and mutually consistent. This stands in contrast to other work on agents' mental state (see Section 7), which makes further assumptions about agents acting in their own best interests, and so on. Such stronger notions of rationality are obviously important, and in the future we may wish to add them. However, neither the concept of agenthood nor the utility of agent-oriented programming depend on them.

Although I discuss formal definitions, in the presentation here I leave out many details, inclusion of which would defeat the purpose of this document. In particular, I have omitted almost all discussion of formal semantics (I have left in only terse outlines, for the benefit of the readers who particularly care about such matters; others may skip those few and short segments).

[5]The material in this section summarizes work carried out jointly with Becky Thomas, Anton Schwartz and Fangzhen Lin

4.1 Time

The basis for our language is a simple temporal language. The particular temporal logic we adopt here is an explicit-time point-based logic; the construction would be similar for other choices. We take time to be a linear order. In fact, here we will assume the integers as a model for time; this too is a convenience one might decide to do without later. A 'time line' associates with each time point a set of propositions, those that are true at that time.

Syntax

Assume a set TC of time point constants, a set P of predicate symbols (each with a fixed arity ≥ 0), a set F of function symbols (each with a fixed arity ≥ 0), and set V of variables. The set of terms is defined as follows: a variable is a term, and if f is an n-ary function symbol and $\mathtt{trm}_1, \ldots, \mathtt{trm}_n$ are terms then $\mathtt{f}(\mathtt{trm}_1, \ldots, \mathtt{trm}_n)$ is also a term. The set of well-formed formulas is then defined as follows. If $t_1, t_2 \in TC$ then $t_1 < t_2$ is a wff. If $\mathtt{t}$ is a time point symbol, r is an n-ary predicate symbol, and $\mathtt{trm}_1, \ldots, \mathtt{trm}_n$ are terms, then $\mathtt{f}(\mathtt{trm}_1, \ldots, \mathtt{trm}_n)^{\mathtt{t}}$ is a wff. If φ and ψ are wffs and v is a variable, then $\varphi \wedge \psi$, $\neg\varphi$ and $\forall v \varphi$ are all wffs. We will use the standard abbreviations $\vee$ and $\supset$. (In this simple construction we do not even allow quantification over time points.)

Semantics (outline)

The models for sentences are conventional: time point constants are interpreted as integers[6], variables and 0-ary function symbols as objects, other function symbols as functions from time and objects to objects, and predicate symbols as functions from to time to tuples of objects. If M is such an interpretation, then $\mathtt{P}(\mathtt{trm}_1 \ldots \mathtt{trm}_n)^{\mathtt{t}}$ is true in it iff $(M(\mathtt{t},\mathtt{trm}_1), \ldots, M(\mathtt{t},\mathtt{trm}_n)) \in M(\mathtt{t},\mathtt{P})$.

4.2 Belief

We now augment the language with a modal operator B, denoting belief. The standard logic of belief (see [23]) simply adds to the language the sentence $\mathtt{B}\varphi$, for any sentence φ in the

[6] In this document we assume the integers as the temporal structure, but one could assume otherwise.

language. The most common logic of belief is the KD45 system (see [7]), which in addition to the axioms of propositional calculus and two inference rules satisfies the following axioms:

$\mathtt{B}\varphi_1 \wedge \mathtt{B}(\varphi_1 \supset \varphi_2) \supset \mathtt{B}\varphi_2$

$\mathtt{B}\varphi \equiv \mathtt{BB}\varphi$

$\neg\mathtt{B}\varphi \equiv \mathtt{B}\neg\mathtt{B}\varphi$

$\neg\mathtt{B}(\varphi \wedge \neg\varphi)$

We will complicate the logic in only a minor way by adding to the operator two more arguments: the agent who is doing the believing, and the time of belief. Furthermore, possible worlds will also have a temporal dimension, and will in fact each be a time line.

Syntax

Given a temporal logic as before, we assume another set of constants AG, the agent constants. We now add one wff-formation rule: if t is a time-point constant, a is an agent constant, and φ is a sentence in the language, then $\mathtt{B}^{\mathtt{t}}(\mathtt{a},\varphi)$ is also a sentence in the language. For example, $\mathtt{B}^{3}(\mathtt{a},\mathtt{B}^{10}(\mathtt{b},\mathtt{like}(\mathtt{a},\mathtt{b})^{7}))$ will mean that at time 3 agent a believes that at time 10 agent b will believe that at time 7 a liked b.

We assume B to be an KD45 operator; among other properties this includes:

$\mathtt{B}^{\mathtt{t}}(\mathtt{a},\varphi_1) \wedge \mathtt{B}^{\mathtt{t}}(\mathtt{a},\varphi_1 \supset \varphi_2) \supset \mathtt{B}^{\mathtt{t}}(\mathtt{a},\varphi_2)$

$\mathtt{B}^{\mathtt{t}}(\mathtt{a},\varphi) \supset \mathtt{B}^{\mathtt{t}}(\mathtt{a},\mathtt{B}^{\mathtt{t}}(\mathtt{a},\varphi))$

$\neg\mathtt{B}^{\mathtt{t}}(\mathtt{a},\varphi) \supset \mathtt{B}^{\mathtt{t}}(\mathtt{a},\neg\mathtt{B}^{\mathtt{t}}(\mathtt{a},\varphi))$

$\neg\mathtt{B}^{\mathtt{t}}(\mathtt{a},\varphi \wedge \neg\varphi)$

Semantics (outline)

A B-structure is a tuple (L, m) where L is a set of time lines, and m is a function that specifies for every time point and agent an accessibility relation $R_{B_{t,a}}$ on L. Each such accessibility relation is transitive and Euclidean, and B is defined to be the necessity operator for this modality.

4.3 Belief versus knowledge

In this document I will not assume that agents possess knowledge that is distinct from their beliefs. The distinction usually made between the two is that while known facts must in fact be true, believed facts need not be. It is possible to add another modality K to represent knowledge, with the extra property $\mathtt{K}^t(\mathtt{a},\varphi) \supset \varphi$. In fact, together with Y. Moses I have made a proposal to view belief as a defeasible form of knowledge, and actually defined the B operator in terms of K; see details in [50]. However, since on the one hand the proposal is sufficiently novel to attract controversy, and on the other hand one can define a coherent and useful notion of mental state without distinguishing between knowledge and belief, I will not pursue this issue here.

4.4 Commitment

So far I have used largely well-known constructions: the temporal logic is standard, the logic of belief is standard, and their combination, although somewhat novel, is nonetheless straightforward. We now depart more radically from past constructions and introduce new modal operator, CMT.

Unlike B, CMT is a ternary operator: $\mathtt{CMT}(\mathtt{a},\mathtt{b},\varphi)$ will mean that agent a is committed to agent b about φ (of course, we will add a temporal component to represent the time of commitment). This inter-agent flavor of commitment contrasts with past accounts of the same concept, which viewed it as an intra-agent phenomenon. Notice that the agent is committed about the truth of a sentence, not about his taking action. In fact, I will not introduce in the logic a seperate category of entities called "action." For example, strictly speaking, rather than say that the robot is committed to taking the action "raise arm" at time t, we will say that the robot is committed to the proposition "the robot raises its arm at time t." However, since actions are such a natural concept and in fact are dealt with in a special way, in the actual programming language we will introduce them as syntactic sugar. Both in the logical treatment here and in the programming language later, since actions are facts, they are also instantaneous (we have adopted for now a point-based logic). In order to represent durational actions, we must currently break them into consecutive instantaneous

one; it will be important, and not too hard, to avoid this in future extensions.

Syntax

We augment the syntax of the language as follows: If a and b are agent terms, t is a temporal term and φ is a sentence, then $\mathtt{CMT^t(a,b,\varphi)}$ is also a sentence.

We take CMT to be a KD4-operator (see [7]); among other properties, this gives us the following:

$$\mathtt{CMT^t(a,b,\varphi_1)} \wedge \mathtt{CMT^t(a,b,\varphi_1 \supset \varphi_2)} \supset \mathtt{CMT^t(a,b,\varphi_2)}$$

$$\mathtt{CMT^t(a,b,\varphi)} \supset \mathtt{CMT^t(a,b,CMT^t(a,b,\varphi))}$$

$$\neg\mathtt{CMT^t(a,b,\varphi \wedge \neg\varphi)}$$

Semantics (outline)

A B-CMT-structure is the result of adding to a B-structure a second function, which specifies for each time point t and each ordered pair of agents a, b a second accessibility relation on time lines, $R_{CMT_{t,a,b}}$. Each such accessibility relation is transitive and serial, and CMT is defined to be the necessity operator of this modality.

4.5 Belief and commitment

Beliefs and commitment must not only be internally consistent, they must also be mutually consistent. First, we assume that an agent is completely aware of his commitments:

$$\mathtt{CMT^t(a,b,\varphi)} \equiv \mathtt{B^t(a,CMT^t(a,b,\varphi))}$$

$$\neg\mathtt{CMT^t(a,b,\varphi)} \equiv \mathtt{B^t(a,\neg CMT^t(a,b,\varphi))}$$

Note that an agent is not neccesarily aware of which coomitment are made to him, only of commitment he has towards others. Second, we assume that agents only commit in good faith:

$$\mathtt{CMT^t(a,b,\varphi)} \supset \mathtt{B^t(a,\varphi)}$$

Note that as a corollary we have that commitments must be consistent:

Fact: $\mathtt{CMT^t(a,b,\varphi)} \supset \neg\mathtt{CMT^t(a,c,\neg\varphi)}$

4.6 Choice

The freedom to choose among several possible actions is central to the notion of agenthood,[7] and earlier on in the research we indeed took choice to be a primitive notion. The current definition of commitment provides an alternative, however: choice is defined to be simply commitment to oneself:

$$\mathtt{CH}^t(\mathtt{a},\varphi) =_{def} \mathtt{CMT}^t(\mathtt{a},\mathtt{a},\varphi)$$

It should be added that, as is the case in general in language, the word 'choice' is multi-facetted, and it is not our aim to capture all senses of the word. The sense of choice here is akin to that of 'decision'; an agent has chosen something if he has decided that that something be true. In particular, no connection is assumed here to any notion of motivation, such as desire or intention. See related discussion in subsection 4.9.

4.7 Capability

Also intimately bound to the notion of agenthood is that of capability. I may choose to move my arm, but if I am not capable of it then it will not move. I will not ask a three-year-old, nor a mobile robot, to climb a ladder, since I do not believe they are capable of it.

Unlike the notions discussed so far, capability is not a purely internal property of the agent. In fact, there are philosophical views which completely dissociate capability from mental state (e.g., [15]. We, however, choose to view the notion as a certain relation between the agent's mental state and the world. One could introduce an independent operator to denote capability, but we have decided to define it away. The intuition behind the following definition is that 'to be able to X' means to have the power to make X true by merely choosing that it be so:

$$\mathtt{CAN}^t(\mathtt{a},\varphi) =_{def} \mathtt{CH}^t(\mathtt{a},\varphi) \supset \varphi$$

[7] To quote Isaac Bashevis-Singer, "We *must* believe in free will; you see, we have no choice."

To be sure, this definition departs from common sense quite sharply on certain points, but so far we have not found ourselves hurt by this gap. For example, although we get that $\neg CH^t(a,\varphi) \supset CAN^t(a,\varphi)$, (i.e., one is capable of anything by merely not choosing it), that turns out to be quite innocuous; what count are the choices that are *not* ruled out. And while $\varphi \supset CAN^t(a,\varphi)$ (i.e., one is capable of anything that happens to be true), the typical statements refer to beliefs about capabilities, such as $B^t(a,CAN^{t'}(b,\varphi))$, which is not entailed by φ. Of course, we do get $B^t(b,\varphi) \supset B^t(b,CAN^t(a,\varphi))$, but we see no harm in assuming that anyone can bring about something that we believe will happen anyway.[8]

The definition does have some very intuitive properties, such as the following

Fact: $CMT^t(a,b,\varphi) \supset B^t(a,CAN^t(a,\varphi))$

As with the previous notions we are open to other definitions, but until we encounter difficulties we intend to adopt the simplest systems. Again, the game we are playing is not to determine whether a particular definition can be shown to contradict common sense or linguistic convention, but whether restricted definitions will support significant applications.

4.8 The persistence of mental states

So far all the restrictions on mental attitudes referred to attitudes at a single instant of time. We conclude the discussion of mental state by discussing restrictions on how mental states change over time.

Consider, for example, belief. Our axioms so far allow models in which at one time an agent believes no propositional sentences but tautologies, at the next time he has a belief about *every* sentence, and at the time following that he is again very agnostic . We have the intuition that beliefs tend to be more stable than that. We will now place a strong condition on belief; we will assume that agents have perfect memory of and faith in their beliefs, and only let go of a belief if they learn a contradictory fact. Beliefs therefore persist *by default.* Furthermore, we will assume that the *absence* of belief also persists by default, although in

[8]It is possible to add a necessity operator in the definition, as in $CAN^t(a,\varphi) =_{def} \Box(CH^t(a,\varphi) \supset \varphi)$, where $\Box$ is interpreted as truth in all worlds. However, while there is a better match with intuition under this definition, we have so far found no other advantage to it, and so for now we have decided against this more complex definition.

a slightly different sense: if an agent does not believe a fact at a certain time (as opposed to believing the negation of the fact), then the only reason he will come to believe it is if he learns it. (In a forthcoming publication, where we consider also the persistence of knowledge and ignorance, we assume that the persistence of knowledge is absolute: what you know now you always will. The persistence of ignorance is, again, only by default.)

How to formally capture these two kinds of default persistence is another story, and touches on issues that are painfully familiar to researchers in nonmonotonic temporal reasoning and belief revision. In fact, a close look at the logical details of belief (or knowledge) persistence reveals several very subtle phenomena, which have so far not been addressed in the literature. I will use the following seemingly-formal sentences:

$$\mathtt{B}^{t}(a,\varphi) \wedge \neg\mathtt{LEARN}^{t}(a,\neg\varphi) \supset \mathtt{B}^{t+1}(a,\varphi)$$

$$\neg\mathtt{B}^{t}(a,\varphi) \wedge \neg\mathtt{LEARN}^{t}(a,\varphi) \supset \neg\mathtt{B}^{t+1}(a,\varphi)$$

However much more needs to be added in order for the right persistence to take place; a more detailed treatment will appear elsewhere.

Commitments too should persist; they wouldn't be commitments otherwise. As in the case of belief, however, the persistence is not absolute. Although by default commitments persist, there are conditions under which commitments are revoked. These conditions presumably include explicit release of the committer by the committee, or alternatively a realization on the part of the committer of the impossibility of the commitment. (Cohen and Levesque [8] actually propose a more elaborate second condition, one that requires common knowledge by the committer and committee of the impossibility; however further discussion of their position and arguments against it would be too long a detour; see a brief discussion of their work in subsection 4.9.)

I will use the sentence

$$\mathtt{CMT}^{t}(a,b,\varphi) \wedge \neg\mathtt{REVOKE}^{t}(a,b,\varphi) \supset \mathtt{CMT}^{t+1}(a,b,\varphi)$$

but again I caution that more needs to be done in order to preclude unwarranted revoking of a commitment.

Since choice is defined in terms of commitment, it inherits the default persistence. Notice, however, an interesting point about the persistence of choice: while an agent cannot revoke

commitments he made to others, he can cancel commitments that were made to him – including commitments he made to himself, namely choices. An agent can therefore freely modify his choices.

Finally, capabilities too tend to not fluctuate wildly. In fact, in this document I assume that capabilities are fixed: what an agent can do at one time it can do at any other time. However, I will allow to condition a capability of an action on certain conditions that hold at the time of action.

4.9 A short detour: Comparison with Cohen and Levesque

In several publications (e.g., [9, 8]) Cohen, Levesque and several associates have investigated the logical relationships between several modalities such as the above-mentioned ones. As mentioned earlier I have deferred discussion of related work to a later section. However, since there is room for confusion between Cohen and Levesque's definitions of mental categories and our own, I will make an exception in this case. The reader may skip this subsection, although I believe the discussion may provide further intuition about our definition as well as Cohen and Levesque's.

Cohen and Levesque employ two basic modalities, BEL (belief) and G (goal, or choice). Although these bear a resemblance to our belief and choice modalities, important differences exist. Their belief modality is really the same as ours, except that they are not as explicit about its temporal aspect. If I understand their construction correctly, one may use the $\Diamond$ tense operator to specify that "sometime in the future an agent will believe a fact, either about that time or about a time yet further into the future," but one is not able to talk about (e.g.) the agent believing in the future something about the past. This capability could be achieved by adding other tense operators, if the authors insisted on a tense logic.[9]

The primary intuition offered about the G modality is that it denotes choice ("Consider the desire that the agent has chosen to pursue as put into a new category. Call this chosen desire, loosely, a goal"). However, I have already noted that term 'choice' is multi-facetted, and indeed G is quite different from our CH. To start with a technical distinction, whereas CH is a KD4 system, G does not have the "positive iteration" property; that is,

[9]However, the authors do find it necessary to mention explicit dates, which they represent by propositions such as '1/1/90/12:00'. That being the case, I do not see the the utility of retaining the tense operators.

(G a (G a p)) ≡ (G a p) is not an axiom. The reason for this is that for Cohen and Levesque 'choice' contains an element of desire. The choices of an agent, for Cohen and Levesque, constitute a consistent subset of the agent's desires, those that the agent has adopted as goals. In contrast, our CMT modality (and hence the derived CH modality) reflects absolutely no motivation of the agent, and merely describes the actions to which the agent is committed.

The difference in senses is the difference between a decision to act and a decision to pursue a goal. This difference is reflected in the different interactions between choice and belief. In our construction, commitment (and therefore also choice) implies belief: CH(a,p)⊃B(a,p) (if an agent decides on an action he believes it will take place). The converse implication does not hold in our construction (the agent may believe that the sun will rise tomorrow without making a choice in this regard). For Cohen and Levesque, on the other hand, belief does imply choice: (BEL a p) ⊃ (G a p) (see, e.g., their Proposition 17). The intuition here is that the G modality specifies possible worlds chosen by the agent, and these worlds are selected among the ones the agent believes are possible; therefore if a fact is true in all worlds believed possible by the agent, it must be true in the subset he selects. Note that both senses of choice guarantee that an agent does not choose something he believes impossible: both CH(a,p)⊃¬B(a,¬p) and (G a p) ⊃ ¬(BEL a ¬p) are theorems in the respective systems.

Both senses of choice are worthwhile, although one can imagine other ways of capturing a decision to pursue a goal. In particular, I am not sure Cohen and Levesque made the best decision when they decided to use a single operator to capture both 'having a goal' and 'deciding to adopt a goal.'[10] For example, it may prove advantageous to start with a G modality denoting 'having a goal,' and define 'goal adoption' by CH(a,G(a,p)) (with the appropriate temporal arguments added). However, I have deliberately avoided such more complex notions in this document as they are not needed for the fundamentals of AOP, and will not pursue the issue further.

Finally, Cohen and Levesque define the notion of persistent goals, which intuitively are

[10]Notice that the issue does not arise in the cases of belief and (our sense of) commitment. With regard to belief, it makes no sense to commit to believing something, at least not in the sense of knowledge-like belief that we have been discussing. With regard to commitment, it does make sense to commit to be being committed, but, by our definition, that is identical to committing.

goals that are retained over time until the agent believe that either they were achieved or are impossible.[11] This treatment is closer in spirit to our treatment of the persistence of mental attitudes. However, Cohen and Levesque only state that either the goal persists or else the particular beliefs occur, and do not provide a mechanism by which goals indeed persist by default, and spurious beliefs do not occur. As mentioned in our treatment, this is a nontrivial task.

It should be added that, although I have been critical of a few of their choices, our treatment of mental state has benefitted much from their work. Furthermore, they have tackled complex notions such as goals, desires and intentions, which I have not yet dared.

5 The programming of agents[12]

In the previous section I discussed the first component of the AOP framework, namely the definition of agents. I will now discuss the second component, the programming of agents.

The behavior agents is governed by programs; each agent is controlled by his own, private program. Agent programs are in many respects similar to standard programs, containing primitive operations, control structures and input-output instructions. What makes them unique is that the control structures refer to the mental-state constructs defined previously, and that the IO commands include methods for communicating with other agents. We do not yet have a running agent interpreter.[13] I will instead describe here a hypothetical agent language, called AGENT0. AGENT0 embodies many simplifying assumptions, but still it illustrates many issues in agent-oriented programming.

In a conventional programming language the syntax defines the primitive operations available, and the programmer specifies which of those are to be carried out and in which order. This much is true of AGENT0 as well. In regular languages, however, there is a simple mapping between the structure of the program and the order of execution; typically, a linear

[11] Cohen and Levesque use the clause (BEL x p) $\vee$ (BEL x $\Box\neg$p). It would seem that their intuition would support the weaker condition (BEL x (p $\vee$ $\Box\neg$p)).

[12] This section describes work that has benefitted from a collaboration with Jun-ichi Akahani

[13] However, J. Akahani has been experimenting with some prototypes. His approach is interesting in that it is in the style of logic programming, which is to say that program statements are themselves logical sentences. I myself will not assume that agent programs are themselves part of a logical language, only that they make use of the logic for describing mental state.

sequence of commands translates to the same execution order. In contrast, in AGENT0 there is complete decoupling between the time a command is issued (and in particular the order between different commands) and the time at which it is executed; at any time a commitment can be made about any future time. In fact, in AGENT0 an agent is continually engaged in two types of activity: making commitments about the future (and, as a special case, making choices), and honoring previous commitments whose execution time has come (and which have not been revoked in the meanwhile). In the following subsections I describe the syntax of AGENT0 and its interpreter.

A word about the relationship between the logical development in the previous section and the programming language. The logical treatment provided quite general constraints on the mental states of agents. AGENT0, on the other hand, will represent mental state very concretely, imposing very strong additional constraints (for example, beliefs will all be atomic sentences). As one reads about AGENT0 one may wonder what role is served by the general treatment in the previous section.

The answer is that the general constraints allow one to guarantee various properties of agents, independent of the implementation; the properties are guaranteed as long as the implementation meets the general constraints, which AGENT0 does. Every implementation will then have additional properties, and indeed in the simple-minded AGENT0 many special properties can be shown, but those all lie within the general constraints of agenthood.

5.1 The syntax of AGENT0

In the programming language itself one specifies only conditions for making commitments; commitments are actually made and, later, actually carried out, automatically at the appropriate times (see discussion of the interpreter in Subsection 5.2 below). Before we define the syntax of commitments we need a few preliminary definitions. I will first develop the syntax of AGENT0 in a bottom-up fashion, and then summarize it in BNF notation. The reader may wish to refer forward to the formal definition while reading the following description.

Fact statements

Fact statements are fundamental to AGENT0; they are used to specify the content of actions as well as conditions for their execution. Fact statements are simply sentences in the

temporal-modal logic developed in the previous section in which the mental states of agents were defined. They can be simple facts about the world, such as `[t,employee(smith,acme)]`, or may refer to the mental state of agents, as in
`[t',¬B(a,[t,employee(smith,acme)])]`. (AGENT0 does not boast a friendly syntax; in later versions a more congenial syntax will be adopted.) When I discuss the interpreter in Section 5.2 it will become clear that we will want to restrict facts to some subset of the language for efficiency reasons, but I will not pursue that issue further here.

Unconditional action statements

Agents commit to action, and so we now specify what actions are. We make two orthogonal distinctions between types of action: actions may be *private* or *communicative*, and, independently, they may be *conditional* or *unconditional.*

The syntax for private actions is

```
DO(t,p-action)
```

where `t` is a time point and `p-action` is a private action name. Private action names are idiosyncratic and unconstrained; a data base agent may have retrieval primitives, a statistical computation agent may run certain mathematical procedures, and a robot may servo itself. Private actions are analogous to the implementation of specific methods in OOP. Private actions may be invisible to other agents, as in the data base example, but need not be so, as in the robot example.

Private actions may or may not involve IO. Communicative actions, on the other hand, always involve IO. Unlike private actions, communicative actions are uniform, and common to all agents. While in a general AOP system we can expect many types of communicative action, the restricted version AGENT0 has only three types of communicative action: informing, requesting, and cancelling a request.

The syntax of informing is

```
INFORM(t,a,fact)
```

where t is a time point, a is an agent name and fact is a fact statement. Note that t is the time at which the informing is to take place, and fact itself contains other temporal information, as in INFORM(5,b,[1,employee(smith,acme)]).[14]

The syntax of requesting is

```
REQUEST(t,a,action)
```

where t is a time point, a is an agent name and action is an action statement, defined recursively. So, for example, REQUEST(1,a,DO(10,update-database)) constitutes a legitimate request. Again, one should distinguish between the time at which the requesting is to be done (1, in this example) and the time of the requested action (10, in the example). Requests can be embedded further, as in REQUEST(1,a,REQUEST(5,b,INFORM(10,c,fact))).

The syntax of cancelling a request is:

```
UNREQUEST(t,a,action)
```

where t is a time point, a is an agent name and action is an action statement.

The last unconditional action in AGENT0 is really a nonaction. Its syntax is:

```
REFRAIN action
```

where action is an action statement which does not itself contain a REFRAIN. The role of refraining will be to prevent commitment to other actions.

Conditional action statements

In AGENT0 we distinguish between commitments for conditional actions, which include conditions to be tested right before acting, and conditions for making the commitment to act in the first place. I now discuss only conditional actions, and will discuss conditions for making commitments later. Conditional actions rely on one form of condition, called a *mental condition*, while conditions for making a commitment include both mental conditions

[14] As is again obvious, attractive syntax is not the main advantage of AGENT0.

and so-called *message conditions*. The tests for these two kinds of condition constitute AGENT0's control structures. I now discuss only mental conditions.

Mental conditions refer to the mental state of the agent, and the intuition behind them is that when the time comes to execute the action, the mental state *at that time* will be examined to see whether the mental condition is satisfied. For this reason the agent- and time-components of the mental state are implicit and can be omitted in the specification of mental conditions. A mental condition is thus any combination of modal statements in the temporal-modal language, with the primary 'agent' and 'time' arguments omitted.

Specifically, a mental condition is a logical combination of *mental patterns*. A mental pattern is one of two pairs:

`(B,fact)` or `((CMT,a),action)`

where `fact` is a fact statement, `a` is an agent name and `action` is an action statement. An example of a mental pattern is `(B,employee(smith,acme))`.

Given the syntax of mental conditions, the syntax of a conditional action is

`IF mntlcond THEN action`

where `mntlcond` is a mental condition and `action` is an action statement. An example of a conditional action is

`IF (B,employee(smith,acme)) THEN INFORM(t,a,employee(smith,acme))`

The intuitive reading of this action is "if at time `t` you believe that `smith` is an employee of `acme`, then at that time inform agent `a` of that fact."

As was said, mental conditions may contain logical connectives; the following three actions together constitute a `QUERY` about whether `fact` is true (`b` is the one being queried, `a` is the one he is asked to inform):

`REQUEST(t,b,IF (B,fact) THEN INFORM(t+1,a,fact) ),`
`REQUEST(t,b,IF (B,NOT fact) THEN INFORM(t+1,a,NOT fact) )`, and
`REQUEST(t,b,IF NOT (BW,fact) THEN INFORM(t+1,a,NOT [t+1,BW(a,fact)]) )`[15]

[15] `BW` is the "believe whether" operator, defined by `[t,BW(a,p)]` $\equiv$ `[t,B(a,p)]` $\vee$ `[t,B(a,`$\neg$`p)]`.

Variables

In the style of logic programming and production systems, in AGENT0 procedures are invoked in a pattern-directed fashion. Specifically, we will see that commitment rules are "activated" based on certain patterns in the incoming messages and current mental state. Variables play a crucial role in these patterns.

A variable is denoted by the prefix '?'.[16] Variables may substitute agent names, fact statements or action statements. Thus the following is a legitimate conditional action:

```
IF NOT ((CMT,?x),REFRAIN action) THEN action
```

In the tradition of logic programming, variables in action statements (including the mental condition part) are interpreted as existentially quantified. The scope of the quantifier is upwards until the scope of the first NOT, or it is the entire statement, if the variable does not lie in the scope of a NOT. Thus the last statement reads informally as "if you are not currently committed to anyone to refrain from action, then take that action."

It is advantageous to allow other quantifiers as well. The one quantifier I will introduce in AGENT0 is a limited form of the universal quantifier, but in the future others, such as the "latest (earliest) time point such that" quantifier, may be introduced. The universally-quantified variables will be denoted by the prefix '?!'. The scope of these variables is always the entire formula, and thus the conditional action

```
IF (B,emp(?!x,acme)) THEN INFORM(a,emp(?!x,acme))
```

results in informing a of all the individuals who the agent believes to be acme employees.

Having discussed action statements, we can now finally discuss the type of statements that actually appear in the program, namely commitment statements.

[16] Although in the actual implementation I intend to use a Prolog-like unification mechanism, since I want to distinguish between built-in constants such as REQUEST and application-dependent ones such as employee, I did no want to reserve (e.g.) upper case letters for variables. At the level of the current discussion these details are, of course, of little significance.

Commitment statements

Since action statements contain information about what needs to be done, about when it needs to be done, and even the preconditions for doing it, one might have expected a collection of action statements to constitute a program. However, there is another crucial layer of abstraction in AGENT0. Most of the action statements are unknown at programming time; they are later communicated by other agents (one of which may be the "user," in situations where that concept is applicable). The program itself merely contains conditions under which the agent will be committed to actions. Some of these conditions may be trivial, resulting in a priori commitments, but most commitments will be in response to messages.

The conditions under which a commitment is made include both mental conditions, discussed above, and message conditions, which refer to the current incoming messages. A message condition is a logical combination of *message patterns*. A message pattern is a triple

```
(From,Type,Content)
```

where `From` is the sender's name, `Type` is `INFORM` or `REQUEST`, and `Content` is a fact statement or an action statement, depending on the type. The other information associated with each incoming message, its destination and arrival time, are implicit in this context and are thus omitted from the message pattern (of course, the `content` will include reference to time, but that is the time of the fact or action, not the arrival time of the message). An example of a message pattern is `(a,INFORM,fact)`, meaning that one of the new messages is from `a` informing the agent of `fact`. An example of a more complex message condition is `(a,REQUEST,DO(t,walk)) AND NOT (?x,REQUEST,DO(t,chew-gum))`, meaning that there is a new message from `a` requesting the agent to `walk`, and there is no new request from anyone that the agent `chew-gum`.

The syntax of a commitment statement is then simply

```
IF MSGCOND: msgcond AND MNTLCOND: mntlcond
    THEN COMMIT TO agent ABOUT action
```

where `msgcond` and `mntlcond` are respectively message and mental conditions, `agent` is an agent name, and `action` is an action statement. Note that the action statement itself may

be conditional, containing its own mental condition. In fact, we will use a more concise and less Cobolish syntax for commitment statements, namely:

```
COMMIT(msgcond,mntlcond,agent,action)
```

An example of a commitment statement is

```
COMMIT(  (?a,REQUEST,?action),
         (B,myfriend(?a)),[17]
         ?a,
         ?action )
```

Finally, a program is simply a sequence of commitment statements, preceded by a definition of the agent's capabilities and initial beliefs.

A BNF description of the AGENT0

Before describing the interpreter for AGENT0 and providing an example, let me summarize the discussion of the syntax by giving its BNF definition. (In accordance with standard conventions, in the following * denotes repetition of zero or more times.)

[17]The reader might have expected other conditions, such as the absence of contradictory prior commitments. However, as is explained below in Subsection 5.2, these conditions are checked for automatically by the interpreter and therefore need not be mentioned explicitly by the programmer.

<program>	::=	TIMEGRAIN[18]:= (a duration in some agreed-upon units: milliseconds, hours/minutes, *etcetera*) CAPABILITIES: (<action>,<mntlcond>)* INITIAL BELIEFS: <fact>)* INITIAL COMMON BELIEFS: (<agent>,<fact>)* COMMITMENT RULES: <commitrule>*
<action>	::=	DO(<time>,<privateaction>) | INFORM(<time>,<agent>,<fact>) | REQUEST(<time>,<agent>,<action>) | UNREQUEST(<time>,<agent>,<action>) | REFRAIN <action> | IF <mntlcond> THEN <action>
<time>	::=	(some representation of time: the integers, *date/hour/minute*, or any other agreed-upon format) | <variable>
<privateaction>	::=	<actionname>(<parameter>*) | <variable>
<actionname>	::=	(an alphabetic string)
<parameter>	::=	(an alphanumeric string) | <variable>
<fact>	::=	[<time>,<predicate>(<arg>*)] | [<time>,B(<agent>,<fact>)] | [<time>,CMT(<agent>,<agent>,<action>)] | [<time>,CAN(<agent>,<action>)] | <variable>
<predicate>	::=	(an alphabetic string)
<arg>	::=	(an alphanumeric string) | <variable>[19]
<agent>	::=	(an alphanumeric string) | <variable>
<commitrule>	::=	COMMIT(<msgcond>,<mntlcond>,<agent>,<action>)
<msgcond>	::=	<msgpattern> | NOT <msgcond> | <msgcond> AND <msgcond>
<msgpattern>	::=	(<agent>,INFORM,<fact>) | (<agent>,REQUEST,<action>)
<mntlcond>	::=	<mntlpattern> | NOT <mntlcond> | <mntlcond> AND <mntlcond>
<mntlpattern>	::=	(B,<fact>) | ((CMT,<agent>),<action>)
<variable>	::=	?(alphanumeric string) |?!(alphanumric string)

[18]See discussion of 'time grain' in the next subsection.

[19]Note that there is no syntactic difference between a primitive fact and a private action; see related discussion of 'choice' in Section 4. I have kept the distinction here because at some later version we may want to place different syntactic restrictions on actions and on facts.

5.2 The AGENT0 interpreter

The interpretation of AGENT0 programs is, in principle, quite simple. It consists of iterating the following two steps:

1. Process the incoming messages, updating the beliefs and commitments (recall that we assume that capabilities of agents are fixed);
2. Carry out the commitments for the current time, possibly resulting in further belief change.

In the remainder of this section I will discuss the details of the interpreter. I will first describe the various stages, and then summarize the main features in a diagram.

Assumption about message passing

AGENT0 includes, among other things, communication commands. In order that those be executable I will assume that the platform is a capable of passing message to other machines addressable by name. AGENT0 itself will define the form and content of these messages.

Assumption about the clock

Central to the operation of the interpreter is the existence of a clock. The main role of the clock is to initiate iterations of the two-step loop at regular intervals (every 10 milliseconds, every hour, etcetera); the length of these intervals, called the 'time grain,' is determined by the settable variable `TIMEGRAIN`. I do not discuss the implementation of such a clock, which will vary among platforms, and simply assume that it exists. We also assume a variable `NOW`, whose value is set by the clock to the current time in the format defined in the programming language (an integer, *date:hour:minute*, etcetera).

In the remainder of the description we make the very strong assumption that a single iteration through the loop lasts less than the time grain; in future versions of the language we will relax this assumption, and correspondingly will complicate the details of the loop itself.

Updating beliefs and commitments

Having specified when each iteration takes place, we now specify what happens in each of the two steps of the iteration. The first step consists of two substeps:

1a. Update the beliefs.

1b. Update the commitments.

To discuss these steps we first need to discuss the representation of beliefs, commitments and capabilities. In AGENT0 they are each represented by a data base. The belief data base is updated as a result of either as a result of being informed or as a result of taking a private action; here we discuss only the former update. In AGENT0 agents are completely gullible: they incorporate any fact of which they are informed, retracting previous beliefs if necessary. (This is of course an extreme form of belief revision, and future versions will incorporate a more sophisticated model; see discussion in the final section.) As the result of being informed the agent not only believes the fact, he also believes that the informer believes it, that the informer believes that it (the agent) believes it, and so on. In fact, as the result of informing, the informer and agent achieve so-called *common belief* (the infinite conjunction of "I believe," "I believe that you believe," etcetera). The belief data base will therefore include private beliefs, represented as simple facts, as well as common beliefs, represented by pairs `(a,fact)` (where `a` is the other party).

Items in the data base of capabilities are pairs `(privateaction,mntlcond)`. The mental condition part allows one to prevent commitment to incompatible actions, each of which might on its own be possible. An example of an item in the capability data base is

```
([!?time,rotate(?!degree1)],
 NOT (CMT(?!time,rotate(?degree2)) AND B(NOT ?!degree1=?degree2 ))).
```

Items in the data base of commitments are simply pairs `(agent,action)` (the agent to which the commitment was made and the content of the commitment).

The algorithm for message-induced belief update consists of repeating the following steps for each new incoming `INFORM` message from agent `a` informing of `fact`:

- Add `(a,fact)` to the belief data base;

- If `fact` is inconsistent with the previous beliefs then modify the old beliefs so as to restore consistency.

This last step is of course potentially complicated; both the check for consistency and the restoring of consistency can in general be quite costly, and in general there will be more than one way to restore consistency. We will assume here that there exist sufficient syntactic restrictions on beliefs so as to avoid these problem. An extreme restriction, though one that still admits many interesting applications, would be to disallow in beliefs any connective other than negation; in this case both consistency check and consistency restoring require at most a linear search of the data base, and much less if a clever hashing scheme is used. Other less extreme restrictions are also possible, but I will not pursue this issue further.

Belief change may lead the agent to revoke previous commitments. One reason is that the original commitment relied, among other things, on certain mental conditions stated in the program. These may have included belief conditions that have now changed. Nevertheless, while it would be a natural addition in future versions, in AGENT0 the interpreter is not assigned the responsibility of keeping track of the motivation behind each commitment; that would require a data-dependency mechanism that I would rather not incorporate yet. However, while *motivation* is not kept track of, *capability* is. Belief change may remove capabilities, since the capability of each private action depends on mental preconditions. And thus whenever a belief update occurs, the AGENT0 interpreter examines the current commitments to private action, removes those whose preconditions in the capability data base have been violated, and adds a commitment to immediately inform the agents to whom he was committed of this development. Exhaustive examinations of all current commitments upon a belief change can be avoided through intelligent indexing, but I will not pursue this optimization issue.

Note that the belief update is independent of the program. The update of commitments, however, depends on the program very strongly. The algorithm for updating the commitments consists of two steps:

- For all incoming `UNREQUEST` messages, remove the corresponding item from the commitment data base; if no such item exists then do nothing.

- Check all program commitment-statement; for each program statement
  `COMMIT(msgcond,mntlcond,a,action)`, if :
  - the message conditions `msgcond` hold of the new incoming message,
  - the mental condition `mntlcond` holds of the current mental state,
  - the agent is currently capable of the `action`, and
  - the agent is not committed to `REFRAIN action`, or, if `action` is itself of the form `REFRAIN action'`, the agent is not committed to `action'`.

  then commit to `a` to perform `action`.

Although I am not explicit about it here, it is clear what it means for the message conditions and mental conditions to hold, given their definitions. An agent is capable of an action under the following conditions:

An agent can request and unrequest anything from anyone.

An agent can inform anyone of a fact he (the agent) believes.

An agent is capable of any private action in the capability data base provided the mental condition associated in the data base with that private action holds at that time,[20]

An agent can refrain from any action that does not itself include refraining, provided he is not already committed to that action.

An agent can perform a conditional action `IF cond THEN action` if he can perform `action` under the condition `cond`.

Carrying out commitments

We have so far discussed the first of the two steps in each iteration of the interpreter, updating the mental state. We now discuss the second step, which is less complex by far.

[20]This last mental condition is separate from the mental condition `mntlcond` mentioned above; the one mentioned above is a condition for making a commitment regardless of whether the agent is capable of the action; in contrast the mental condition currently discussed determines whether the agent is capable of it in the first place.

Recall that each commitment in the commitment data base has a time associated with it: `INFORM(t,a,fact)`, `IF mntlcond THEN DO(t,privateaction)`, etcetera. In this second step the interpreter simply executes all the actions whose time is equal to the value of the variable `NOW`, which, as mentioned at the beginning, is maintained by the clock. The meaning of "execute" depends on the type of action:

`INFORM`: send the appropriate message, and add to the belief data base common belief with the informee of the fact.

`REQUEST` and `UNREQUEST`: send the appropriate message.

`REFRAIN`: no effect on execution (`REFRAIN` commitments only play a role in preventing commitment to other actions).

`DO`: Consulting the belief and commitment data bases, check the mental condition associated in the capability data base with the primitive action; if it holds then perform the primitive action (possibly resulting in a belief update).

`IF-THEN`: Consulting the belief and commitment data bases, test the mental condition; if it holds then (recursively) execute the action.

A flow diagram of the interpreter

The main features of AGENT0's interpreter are summarized pictorially in Figure 2; dashed arrows represent flow of data, solid arrows temporal sequencing.

5.3 A sample program and its interpretation

As an example of AGENT0 programs, consider the flight-reservation scenario described in Section 2. We now present an annotated program implementing the airline representative. Although the scenario was simple to begin with, here we simplify it further by ignoring the exchange relating to the supervisor as well as other aspects of the communication. The idea behind the program is that the relevant activity on the part of the airline is issuing a boarding pass to the passenger, and that confirming a reservation is in fact a commitment to issue a boarding pass at the appropriate time.

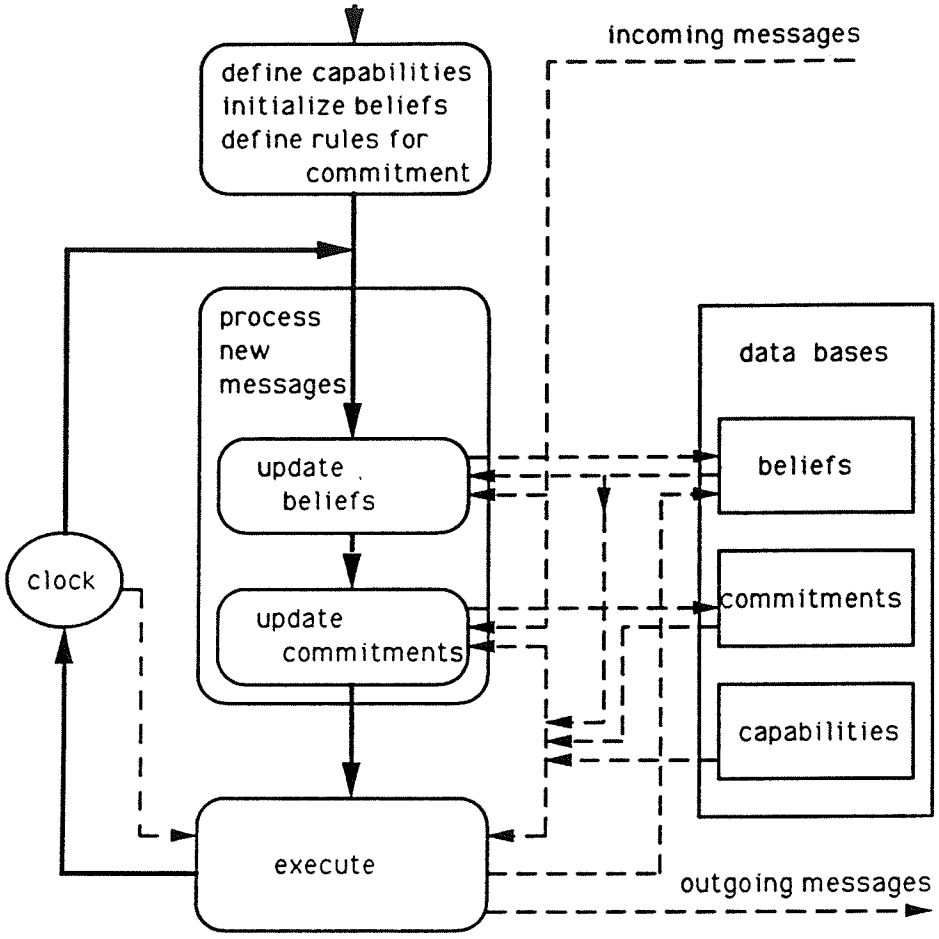

Figure 2: A flow diagram of AGENT0's interpreter

Since some of the low-level definitions are long, it will be convenient to use abbreviations. We will therefore assume that AGENT0 supports the use of macros. We define the following macros:

```
issue_bp(pass,flightnum,date) ⇒
    IF (B,present(pass)) AND B([date/?time],flight(?from,?to,flightnum))
    THEN DO(date/?time-1hr,issue_bp(pass,flightnum,date))
```

Explanation: This no-frills airline issues boarding passes precisely one hour prior to the flight; there are no seat assignments.

```
query_which(t,asker,askee,q) ⇒
    REQUEST(t,askee, IF (B,q) THEN INFORM(t+1,asker,q))
```

Explanation: `query_which` requests only a positive answer; if `q` contains a universally-quantified variable then `query_which` requests to be informed of all instances of the answer to the query `q`.

```
query_whether(t,asker,askee,q) ⇒
    REQUEST(t,askee, IF (B,q) THEN INFORM(t+1,asker,q))
    REQUEST(t,askee, IF (B,NOT q) THEN INFORM(t+1,asker,NOT q))
```

Explanation: `query_whether` expects either a confirmation or a discomfirmation of a fact. It is usually a bad idea to include in the fact a universally-quantified variable.

We now define the airline agent. To do so we need to define its initial beliefs, capabilities, and commitment rules.

Of the initial beliefs, the ones relevant here refer to the flight schedule, and the capacity of each flight. The former are represented in the form `[date/time,flight(from,to,number)]` (ignoring the fact that in practice airlines have a more-or-less fixed weekly schedule), and the latter in the form `[date,capacity(flight,number)]`.

The capability relevant here is that issuing boarding passes. Thus the capability data base contains a single item:

```
issue_bp(?a,?flight,?date),
    (B,[?date,capacity(?flight,?N)]) AND
    (B,?N<|{a:((CMT,?a),issue_bp(?pass,?flight,?date))|})
```

Explanation: issue_bp is a private action involving some external events such as printing a boarding pass and presenting it to the passanger. The |...| denotes cardinality.

Finally, the airline agent has two commitment rules:

```
COMMIT(  (?pass,REQUEST,IF (B,?p) THEN INFORM(?t,?pass,?p)),
         (B,?p),
         ?pass,
         IF (B,?p) THEN INFORM(?t,?pass,?p) )

COMMIT(  (?pass,REQUEST,issue_bp(?pass,?flight,?date)),
         NOT((CMT,?pass),issue_bp(?pass,?anyflight,?date)),
         ?pass,
         issue_bp(?pass,?flight,?date) )
```

In a more realistic example one would have other commitment rules, notifying the the passenger whether his reservation was confirmed, and the reasons for rejecting it in case it was not accepted. In the current implementation the passenger must query that separately.

This concludes the definition of the simple airline agent. Below is a sample exchange between a passenger, smith, and the airline agent. The messages from the passenger are determined by him; the actions of the airline are initiated by the agent interpreter in response.

agent	action
smith	query_which(1march/1:00,smith,airline, [18april/?!time,flight(sf,ny,?!num)]
airline	INFORM(1march/2:00,smith,[18april/8:30,flight(sf,ny,#354)]
airline	INFORM(1march/2:00,smith,[18april/10:00,flight(sf,ny,#293)]
airline	INFORM(1march/2:00,smith,[18april/ ...
smith	REQUEST(1march/3:00,airline,issue_bp(smith,#354,18april))
smith	query_whether(1march/4:00,smith,airline, CMT(airline,smith,issue_bp(smith,#354,18april))
airline	INFORM(1march/5:00,smith, NOT CMT(airline,smith,issue_bp(smith,#354,18april))
smith	REQUEST(1march/6:00,airline,issue_bp(smith,#293,18april))
smith	query_whether(1march/7:00,smith,airline, CMT(airline,smith,issue_bp(smith,#293,18april))
airline	INFORM(1march/8:00,smith, CMT(airline,smith,issue_bp(smith,#293,18april))
...	
smith	INFORM(18april/9:00,airline,present(smith))
airline	DO(18april/9:00,issue_bp(smith,#293,18april))

6 Compiling agent programs[21]

In the previous section I discussed the second component of the AOP framework, agent programs and their interpretation. In this section I discuss, briefly, the compilation of such programs. The reason for the brevity is that we have made only initial progress on this component so far, and this progress is described in detail elsewhere (see below).

Compilers are usually thought of as optimizations on interpreters. This is *not* necessarily the relationship I see between the interpretation and compilation of agent programs. Instead, I see the agent compiler as a general bridge between the intentional level of agent programs and the low-level machine process.

Of course, the interpreter itself is one such bridge, but it requires a direct mapping between the constructs in the agent language and the machine implementing the agent. In particular it requires explicit representation in the machine of beliefs, commitments and capabilities. When we are the ones creating the agents we indeed have the luxury of incorporating these components into their design, in which case the interpreter is adequate. However, we intend AOP as a framework for controlling and coordinating arbitrary devices, and those – cars, cameras, digital watches – do not come equipped with beliefs and commitment rules.

Even if we were in a position to persuade General Motors, Phillips, Finmeccanica and Matsushita to equip every single product with a mental state, we would be ill-advised to do so. AOP offers a perspective on computation and communication that has its advantages, but it is not proposed as a uniform replacement of other process representations. It would be ridiculous to require that every robot-arm designer augment his differential equations with beliefs, or that the digital-watch design verifier augment finite automata with commitments.

However, releasing the manufacturers from the requirement to supply a mental state creates a gap between the intentional level of agent programs on the one hand, and the mechanistic process representation of a given device on the other hand. The role of the compiler is to bridge this gap.

We inherit this decoupling of the intentional level from the machine level from *situated automata*, introduced by Rosenschein in [45] and further developed by him and Kaelbling

[21]The material in this section includes work carried out jointly with Jean-Francois Lavignon

[46, 27]. In situated automata there is a low-level language for describing the device, and another, high level language for the designer to reason about the device. The compiler takes a program written in the high-level language and produces a description of a device in the low-level language.

Like the 'knowledge based' camp in distributed computation we adopt the insight that intentional notions can be viewed as the designer's way of conceptualizing a device (as was discussed also in the introductory section, in connection with McCarthy's and Dennett's ideas). We depart from situated automata when it comes to the details.

In order to define a compiler we need to specify three elements:

1. The source language
2. The target language
3. The compilation process

Of these we have so far addressed in our research only the first and second items. This is clearly a preliminary stage, and for this reason the present section will be short.

Our source language has already been discussed – it is the AGENT0 language defined in the previous section. By way of contrast, situated automata has had several versions; published versions have included a knowledge operator (K) and a tense operator ($\Diamond$, or "eventually").

The choice of target language is particularly important. The language must on the one hand be sufficiently general to cover arbitrary devices, and on the other hand close enough to the hardware so that any device can be naturally described in it. Many process languages exist - synchronous Boolean circuits with or without delays (the choice of situated automata as a target language), finite automata and Turing machines, and various formalisms aimed at capturing concurrency. Our requirements of the process language included the following:

- Representation of process time, including real-valued durations;
- Equal representation of a machine and of the environment
- Asynchronous processes;

- Multiple levels of abstraction

We found that no existing process models met all requirements, and have developed an alternative process model, called *temporal automata.* The model, developed by Jean-Francois Lavignon and myself, is described in detail elsewhere. In [31] we develop the mathematical model, relate it to existing ones, and illustrate it through examples. In [30] Lavignon describes an implemented simulator, which takes as input a definition of a process in the language (called a temporal automaton), creates an internal representation of the automaton, and simulates its behavior graphically.

Although temporal automata are interesting in their own right further discussion of them in this document would be unmotivated, since at this time we have not yet tackled the compilation process itself. The interested reader is referred to the above-mentioned documents for more details.

7 Related work

Except occasionally, I have so far not discussed related past work. This body of related work is in fact so rich that in this section I will be able to only mention the most closely related work, and briefly at that. I do not discuss again past work on logics of knowledge and belief which AOP extends, since I already did that in the introductory chapter. The following work is ordered in what I see as decreasing relevance to, and overlap with, AOP. The order (or, for that matter, inclusion in the list) reflects no other ranking, nor is it implied that researchers high up on the list would necessarily endorse any part of AOP.

- McCarthy's work on Elephant2000 [36]. This language under development is also based on speech acts, and the airline-reservation scenario I have discussed is due to McCarthy. One issue explored in connection with Elephant2000 is the distinction between illocutionary and perlocutionary specifications, which I have not addressed. In contrast to AOP, in Elephant2000 there is currently no explicit representation of state, mental or otherwise. Conditional statements therefore refer to the history of past communication rather than to the current mental state.

- The Intelligent Communicating Agents project (1987-1988), carried out jointly at Stanford, SRI and Rockwell International (Nilsson, Rosenschein, Cohen, Moore, Appelt, Buckley, and many others). This ambitious project had among its goals the representation of speech acts and connection between the intentional level and the machine level. See discussion of some of the individual work below.

- Cohen and Levesque's work on belief, commitment, intention and coordination [9, 8]. This work was discussed in detail in subsection 4.9. To summarize that discussion, Cohen and Levesque too have investigated the logical relationships between several modalities such as belief and choice. Although they have not approached the topic from a programming-language perspective as I have, they too have been interested in speech acts and mental state as building blocks for coordination and analysis of behavior. Their work has its roots in earlier work in natural language understanding by Allen, Cohen and Perrault [2, 10]. The details in the logical treatment of basic mental categories are different, reflecting slightly different intuitions.

- Rosenschein and Kaelbling's situated automata [45, 46, 27]. I already discussed this work in the previous section. To summarize, it is relevant in connection with the compilation of agent programs; we adopt their idea of decoupling the machine language from the programmer's intentional conceptualization of the machine, but differ on the technical details.

- Winograd and Flores's work on coordination [16]. As a part of their more global project, Winograd and Flores have developed a model of communication in a work environment. They point to the fact that every conversation is governed by some rules, which constrain the actions of the participants: a request must be followed by an accept or a decline, a question by an answer, and so on. Their model of communication is, I believe, that of a finite automaton, with the automaton states corresponding to different states of the conversation. This work seems to me to be a macro theory, in contrast to the micro theory of AOP. Winograd is also investigating the relationship between human and computer communication.

- Nilsson's action nets. ACTNET [42] is a language for computing goal-achieving actions that depends dynamically on sensory and stored data. The ACTNET language is based

on the concept of action networks [41]. An action network is a forest of logical gates that select actions in response to sensory and stored data. The connection to AOP, albeit a weak one, is that some of the wires in the network originate from data-base items marked as 'beliefs' and 'goals'. The maintenance of these data bases is not the job of the action net.

- Genesereth's work on informable agents [20, 19]. Genesereth's interest lies primarily in agents containing declarative knowledge that can be informed of new facts, and that can act on partial plans. In this connection he has investigated also the compilation of declarative plans and information into action commands. Genesereth uses the term 'agents' so as to include also low-level finite-automaton-like constructs. As in our temporal automata, mentioned briefly in the previous section, Genesereth treats an agent and an environment symmetrically.

- Recent work on plan representation and recognition by Kautz, Pollack, Konolige, Litman, Allen and others (e.g., [33, 28, 43, 5]). This literature also addresses the interaction between mental state and action, but it is usually concerned with finer-grained analyses, involving the actual representation of plans, reasoning limitations, and more complex mental notions such as goals, desires and intentions.

8 Discussion

I have described the philosophy behind agent-oriented programming, and progress made towards realizing it – both in terms of formal development and in terms of algorithm design.

This is clearly only a beginning. Beside debugging and fine-tuning the logic and programming language I have presented, the framework can be extended dramatically in a number of directions. Below are some of the directions I intend to explore in the future.

- Mental categories. The language for describing mental state can be augmented to include more complex notions such as desires and intentions, allowing a richer set of communicative commands and more structure on the behavior of agents.

- Groundedness of mental categories. One of the contributions of distributed computation to the formal theory of knowledge is the concrete grounding of the semantics: what were formerly purely formal constructs, possible worlds, became the set of possible 'runs' of the system, given a particular protocol. In my logical development I did not anchor belief and commitment similarly, and it would be satisfying to be able to do so.

- Probability and utility. As in most recent work on knowledge and belief, we have adopted very crisp notions of mental attitude; there is no representation of graded belief or commitment. This stands in contrast to game-theoretic work on rational interaction among agents in economics (e.g., [3, 18]) and AI (e.g., [44]), where uncertainty and utility play a key role. This is a natural direction in which to extend our framework.

- Inheritance and groups. In the analogy between OOP and AOP I did not mention inheritance, a key component of OOP today. In OOP if an object is a specialization of another object then it inherits its methods. One analogous construct in AOP would be 'group agents'; that is a group of agent will itself constitute an agent. If we define the beliefs of this composite agent as the 'common beliefs' of the individual agents and the commitments of the composite agent as the 'common commitments' (yet to be defined) of the individual agents, then mental attitudes of the group are indeed inherited by the individual.

- Persistence of mental states. At the end of section 4 I mentioned that the persistence of mental state poses some challenging formal problems: If I believe that you don't believe x, do I believe that you will not believe in a little while? Do I believe that I will believe that you don't? Will I believe then that you don't? Will I believe then that I believed in the past that you didn't know? Answers to these questions depend on some subtle assumptions.

- Resource limitations. In the definition of the interpreter I assumed that the belief and commitment updates all happened fast enough before the next cycle was to start. While often reasonable, this assumption is violated in many real-time applications. In these cases the manipulation of data structures (such as beliefs) must be shortened or supressed in favor of rapid action. There is much interest nowdays in intelligent real-

time problem solving, including issues such as tradeoff between quality and timeliness. From the agent interpreter's standpoint this means that the belief and commitment update cannot proceed blindly, but must take into account the elapsed time, choosing wisely among mental operations.

- Belief revision. AGENT0 adopts an extreme form of belief revision, accepting all new information. Obviously there are situations that call for more discriminating agents, and in fact there is a rich literature on various forms of belief revision (e.g., [17]), and future versions of the interpreter should explore them.

- Societies. Both the theoretical development of mental categories and the AGENT0 programming language concentrated on a single agent. Indeed, the view promoted was of agents functioning autonomously. However, if a society of agents is to function successfully, some global constraints must be imposed. These include societal conventions as well as social structures (such as hierarchies); both reduce the problem solving required by agents and the communication overhead. There is a rich body of literature on computer societies, examples of which include Minsky's informal Society of Mind metaphor [37], Winograd's studies of societal roles, both human and machine [52], Moses and Tennenholtz's recent discussion of the computational advantages of social laws [40], and Doyle's pioneering work on the relationship between rational psychology and economics [13].

These are some of the directions I intend to explore. I have been conservative so far in the scope of the work, but, I believe, more ambitious explorations will have to depart from a clear and rigorous basis of the kind I have defined. First on my agenda is the definition of the compiler, where, as explained in section 6, only initial progress has been made. Of course, the real validation of AOP will be achieved through successful applications of the framework. We are currently engaged in several applications, including ones in civil engineering, user interfaces and robotics.

Acknowledgements. I have discussed AOP in general and this document in particular with many people, and have benefitted from their comments. Members of the Nobotics group, including Jun-ichi Akahani, Nita Goyal, Jean-Francois Lavignon, Fangzhen Lin, Eyal Moses, Anton Schwartz, Dominique Snyers, Becky Thomas and Mark Torrance have contributed in many ways. I have discussed agents and agenthood also with David Cheriton, Tom Dean, Mike Genesereth, Joe Halpern, Yoram Moses, Barbara Hayes-Roth, Leslie Kaelbling, Nils Nilsson, Stan Rosenschein, Rich Thomason, Brian Williams, Terry Winograd, and many others; I apologize for not mentioning everyone. I thank Phil Cohen, Kurt Konolige and Martha Pollack for critical comments. Finally, special thanks to John McCarthy for enlightening conversations.

References

[1] P. Agre and D. Chapman. Pengi: An implementation of a theory of activity. In *Proceedings 6th AAAI*, pages 268–272, 1987.

[2] J. F. Allen. Recognizing intentions from natural language utterances. In M. Brady and R. C. Berwick, editors, *Computational Models of Discourse*, pages 107–166. MIT Press, Cambridge, MA, 1983.

[3] R. Aumann. Agreeing to disagree. *The Annals of Statistics*, 4:1236–1239, 1976.

[4] J. L. Austin. *How to Do Things with Words.* Harvard University Press, 1955/1975.

[5] M. E. Bratman. *Intention, Plans, and Practical Reason.* Harvard University Press, Cambridge, MA, 1987.

[6] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1), March 1986.

[7] B. F. Chellas. *Modal Logic: An Introduction.* 1980.

[8] P. R. Cohen and H. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(3), 1990.

[9] P. R. Cohen and H. Levesque. Rational interaction as the basis for communication. In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication.* MIT Press, Cambridge, Ma., In press.

[10] P. R. Cohen and C. R. Perrault. Elements of a plan-based theory of speech acts. *Cognitive Science*, 3:177–212, 1979.

[11] T. Dean. *Planning and Control.* (forthcoming book), 1990.

[12] D. C. Dennett. *The Intentional Stance.* MIT Press, Cambridge, MA, 1987.

[13] R. Doyle. Artificial intelligence and rational self-government. Technical Report CMU-CS-88-124, Carnegie-Mellon University, Computer Science Department, 1988.

[14] M. Drummond. Situated control rules. In *Proceedings First International Conference on Knowledge Representation and Reasoning.* Morgan Kaufmann, 1989.

[15] D. Elgesem. He would have done it anyway: the logic of agency, ability and opportunity. Stanford University, CSLI, manuscript, 1990.

[16] F. Flores and T. Winograd. *Understanding Computers and Cognition: a new foundation for design.* Ablex Publishing Corporation, 1986.

[17] P. Gardenfors. *Knowledge in Flux: modeling the dynamics of epistemic states.* MIT Press, Cambridge, MA, 1987.

[18] J. Geanakoplos. Common knowledge, bayesian learning, and market speculation with bounded rationality. mimeo, Yale University, 1988.

[19] M. R. Genesereth. A comparative analysis of some simple architectures for autonomous agents. Technical Report Logic-89-2, Stanford University, Computer Science Department, 1989. also to appear as [21].

[20] M. R. Genesereth. A proposal for research on informable agents. Technical Report Logic-89-4, Stanford University, Computer Science Department, 1989.

[21] M. R. Genesereth. A comparative analysis of some simple architectures for autonomous agents. In K. VanLehn, editor, *Architectures for Intelligence.* Lawrence Erlbaum Associates, Hillsdale, NJ, 1991.

[22] P. Grice. *Studies in the Ways of Words.* Harvard University Press, 1989.

[23] J. Y. Halpern. Using reasoning about knowledge to analyze distributed systems. In J. F. Traub, editor, *Annual Review of Computer Science, Volume 2.* Annual Reviews Inc., 1987.

[24] J. Y. Halpern and Y. Moses. A guide to the modal logics of knowledge and belief: Preliminary draft. In *Proceedings of 9th IJCAI*, pages 480–490, 1985.

[25] B. Hayes-Roth, R. Washington, R. Hewett, M Hewett, and A. Seiver. Intelligent monitoring and control. In *Proceedings 11th IJCAI*, pages 243–249, 1989.

[26] C. Hewitt. Viewing control structures as patterns of passing messages. *Artificial Intelligence*, 8:323–364, 1977.

[27] L. P. Kaelbling. Goals as parallel program specifications. In *Proceedings 7th AAAI*, 1988.

[28] H. A. Kautz. A circumscriptive theory of plan recognition. In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication.* MIT Press, Cambridge, MA, 1990.

[29] K. Konolige. *A Deduction Model of Belief.* Pitman / Morgan Kaufmann, 1986.

[30] J.-F. Lavignon. A simulator for temporal automata. Technical Report in press, Stanford University, Computer Science Department, 1990.

[31] J.-F. Lavignon and Y. Shoham. Temporal automata. Technical Report STAN-CS-90-1325, Stanford University, Computer Science Department, 1990.

[32] H. Levesque. All I know: an abridged report. In *Proceedings of 6th AAAI*, pages 426–431, 1987.

[33] D. J. Litman and J. F. Allen. Discourse processing and commonsense plans. In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication.* MIT Press, Cambridge, Ma., 1990.

[34] W. Litwin. A model for computer life. Stanford University, Computer Science Department, manuscript, 1990.

[35] J. McCarthy. Ascribing mental qualities to machines. Technical Report Memo 326, Stanford AI Lab, 1979.

[36] J. McCarthy. Elephant 2000: A programming language based on speech acts, 1990. unpublished manuscript.

[37] M. Minsky. *The Society of Mind.* Simon and Schuster, 1986.

[38] T. M. Mitchell. Becoming increasingly reactive. In *Proceedings 8th AAAI*, pages 1050–1058, 1990.

[39] R. C. Moore. A formal theory of knowledge and action. In J. R. Hobbs and R. C. Moore, editors, *Formal Theories of the Commonsense World.* Ablex Publishing Corporation, Norwood, New Jersy, 1985.

[40] Y. Moses and M. Tennenholtz. In favor of a society. manuscript, Weizmann Institute of Science, Department of Applied Mathematics, 1990.

[41] N. J. Nilsson. Action networks. In *Proceedings of the Workshop on Planning*, University of Rochester, NY, 1989.

[42] N. J. Nilsson, R. Moore, and M. Torrance. Actnet: an action-network language and its interpreter, 1990.

[43] Martha E. Pollack. Plans as complex mental attitudes. In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication.* MIT Press, Cambridge, MA, 1990.

[44] J. S. Rosenschein and M. R. Genesereth. Deals among rational agents. In *Proceedings 9th IJCAI*, pages 91–99, 1985.

[45] S. J. Rosenschein. Formal theories of knowledge in AI and robotics. Technical Report 362, SRI International, 1985.

[46] S. J. Rosenschein and L. P. Kaelbling. The synthesis of digital machines with provable epistemic properties. In *Proceedings Conference on Theoretical Aspects of Reasoning about Knowledge*, pages 83–86. Morgan Kaufmann, 1986.

[47] M. J. Schoppers. Universal plans for reactive robots in unpredictable environments. In *Proceedings 10th IJCAI*, pages 1039–1046, 1987.

[48] J. R. Searle. *Speech Acts: An Essay in the Philosophy of Language.* Cambridge University Press, Cambridge, 1969.

[49] Y. Shoham. Time for action. In *Proceedings 11th IJCAI*, pages 954–959, 1989.

[50] Y. Shoham and Y. Moses. Belief as defeasible knowledge. In *Proceedings 11th IJCAI*, pages 1168–1172, 1989.

[51] B. Thomas, A. Schwartz, S. Kraus, and Y. Shoham. Preliminary thoughts on an agent description language. *International Journal of Intelligent Systems (to appear)*, 1990.

[52] T. Winograd. A language/action perspective on the design of cooperative work. *Human-Computer Interaction*, 3(1):3–30, 1987-88.