On Agent Design Rationale

Helder Coelho, Luis Antunes and Luis Moniz
INESC
Rua Alves Redol, 9, Ap. 13069, 1000 Lisboa, Portugal
E-mail: {hcoelho, xarax, hal}@inesc.pt

Abstract.
Much of the discussion in Distributed Artificial Intelligence (DAI) is about describing the agent´s functioning or the peculiarities of some problem domain (task). The present paper tries the other way around by approaching the designer´s activity, what alternatives (languages, models, architectures) are considered and what choices are made and why. In point of fact, we want to pick up DAI methodology as our key concern. This is quite important for extensive experimentation around workbenches: how can we evaluate deviations from the expected range of behaviours?
The agent design methodology we discuss has a strong emphasis on the several description levels we consider when modelling a situation. It is important to maintain correctness while travelling from the first sketches of a problem until a final system is completed. Moreover, the same journey can be made backwards and again forwards, to evaluate the results of the experiments we conduct.

1- Introduction.

In a not yet published paper, [Jones 93] commented an earlier version of Shoham's paper on Agent-Oriented Programming [Shoham 93], and he extended his strong criticism to the research carried out in Distributed Artificial Intelligence (DAI) on describing the behaviour of computer systems (so-called agents) in terms of logics of human agency and deliberation. The points put forward are quite relevant to be further studied and thought about, and in our present paper we do our best to clarify the whole space of discussion [Sloman 93].

In general, we observe two main DAI perspectives taken over autonomous agents: (1) a programming perspective following the old ideas of Hewitt's Actor formalism [Hewitt 77, 85] around objects, and (2) an applied modal logic perspective around some class of modalities to characterize their mental states and attitudes [Pörn 70]. Often, other low level perspectives may be introduced, close to the computational implementation of those agents, without any concern on the abstract (formal) inter-relations among the different ways (languages) of describing the mentality or the behaviour of such agents.

However, two separated aspects (dimensions) emerge from agent modelling: the functional and the structural perspectives. In the first one, three kinds of descriptions are quite popular: the one inspired by processes (the oldest one) [Demazeau and Müller 90], the one based on logic [Konolige and Pollack 93], and another one using mental states (with engineering, semantic or pragmatic concerns) [Corrêa and Coelho 93]. In the second perspective, several topics have been studied in a non unified fashion: domain-independent architectures, cooperation, communication and control layers, plan-based and decision-making approaches. Evaluation criteria are also missing along these two dimensions, where common sense intuitions about intentional terms are not clearly separated (or related) from formal demands or computational needs. However, an effort has been done by [Cohen et al 89] in order to set up some order in AI, and the ecology (or MAD: Modelling, Analysis and Design) methodology is a good example to view the agent architecture in function of its behaviour and environment. The ecology triangle is depicted in figure 1, and it is easy to conclude that something is missing.

As benchmarks, testbeds and controlled experimentation are becoming more common in AI [Hanks et al 93], the need for an appropriate methodology for agent design is well recognized in order to evaluate not only theories as interpretations of results of

experimental studies, but also to evaluate designs as the correct translations of formal descriptions. The present paper makes a contribution to clarify these issues, and to open new alleys of research around the most appropriate languages for formal and system engineering work (how can we relate their primitives?). In figure 2 we present a visual diagram of our ideas concerning the dynamics of that triangle along a broad line of design activity.

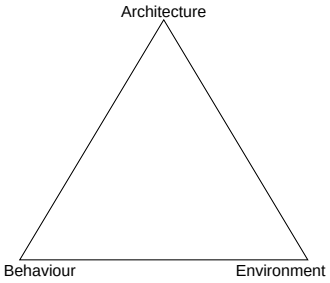

Figure 1: The ecology methodology.

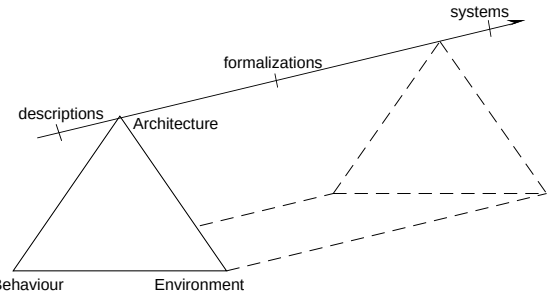

Figure 2: A broad methodology for agent design.

The (forward and backward) movement along a line that connects informal descriptions, formal specifications, systems, programs, and low level code, is also common in informatics (say, information systems), and recent research around OBLOG is giving answers for safer navigation and translation [Sernadas et al 91] along the software life-cycle. Behind such interest is the notion of a software factory, where each translation step is properly measured in order to convince us of its soundness. Debugging systems at the highest level of description (abstraction) is almost analogous to the desire of adopting a natural language for programming, or of pursuing, as in our case, gedanken experiments about social power by adopting only modal-logical techniques.

We defend that the discussion of modelling agents is better supported when set along a 2-D space, where one dimension is responsible for the design level (from descriptions to systems), and the other for the cognitive content (from processes to mental states). Both dimensions may allow the navigation from low to high levels of detail, without fear of loosing rigour or correspondence. By having such a space of reference in background, we can use benchmarks and testbeds in a more principled experimental fashion, weighing up the demands of common sense against those of computational utility in some particular application domains.

Controlled experimentation, in which one varies the features of a multi-agent system or the environment in which it is embedded, and measures the effects of these changes on aspects of social order, is the basis of the science of AI. But, the setting of those experiments, for example on the dynamic aspects of actions in the context of social systems, is aided by using formal approaches to define and describe their organization (the norms for the members of the group) and power structure (what agents can or cannot do in relation to each other). Afterwards, experimentation is guided and better evaluated by theoretical constructs.

2- Processes and (Mental) States.

How can we exhibit all the important aspects of autonomous agents, including data, executable processes, data flow, control flow, and decisions, and all their relationships? How can we simulate the mental behaviour of an agent in particular scenarios? Visualizing different sorts of agents, either from the functional or structural point of view, gives us the understanding about its organization, but seldom at the level of detail mostly required. The mistake of trying to capture the whole picture is only overtaken when we produce revealing little descriptions, which have a photographic exactness about the circumstances of a subject or a scene.

An engineer’s point of view (agent processes architecture) would be more accurate about the functionality of each agent, but again several other points would be missing, such as the aspects of the agent actions (agency, circumstance, and temporality) and the

organizational norms that regulate the agent behaviours. How can we reach a global overview of all these perspectives?

Observing the evolution of the so-called architecture of an agent, in recent years, e.g. from [Demazeau and Müller 90] to [Corrêa and Coelho 93], we may discover the efforts in introducing new graphical design representations for analysis, synthesis and discussion of the agent organizations, and for studying how architectures are related to specific tasks [Sloman 94]. The evolution from a body to a mind, i.e. from function types to mental states, can be understood as a normal way to clarify higher-level processes above the so-called knowledge level [Newell 81]. This shift was necessary when tuning designs to specific tasks (knowledge communication in an intelligent building, the case of Gaspar [Gaspar 90], or conversations, in the case of Corrêa).

In our opinion, seeing agents with the mind's eye, through the notion of mentality, is the best way to look at the information flow from one agent (or situation) to another [Devlin 91]. However, we need to make explicit the bridges between processes and (mental) states to generate sound interpretations about the results obtained. Usually, the literature is full of observations situated on the extremes of the design spectrum [Pollack and Ringuette 90, Genesereth 91], and not much effort has been done to fill the gaps in terminology along that space. So, some misunderstanding is causing confusion, and the discussion is almost impossible (e.g., between Jones and Shoham).

In [Caldas and Coelho 92], we did a first study on evaluating the appropriateness of an agent architecture [Gaspar 90], based upon goals and beliefs, to a specific problem domain, the modelling of oligopolistic economic markets. Along this research we compared two different set-ups, one based upon natural interactions with human beings and another based upon artificial interactions with particular agents. Both performances were evaluated and the conclusions allowed to put forward some deficiencies concerning the kind of mental states selected. However, the graphical apparatus was not sufficient to discover the missing states because we could only see the agent as a black box. As a matter of fact, the metrics for comparing competing systems could only disclose performance differences. By building up a first sketch of a workbench, [Cascalho 93] allowed a deep insight in the workings of an agent's mind, paving the way to new enhancements done later on by [Corrêa 94] or to new tools such as SSAE [Moniz 93a]. We concluded that interesting aspects of system performance were not the sufficient issues to advance our understanding on the inter-relationships between sets of mental states and skills (interact with, reason about, coordinate, identify, relocate) in achieving some task.

The quest for the most suitable functional agent description for conversations was a central topic of the research reported in [Corrêa and Coelho 93]. Again, the departure point was an agent description using only two mental states and some sort of inferential capability (deduction structure), imported from an enhanced and revised proposal of Gaspar's old model [Costa 92]. The specifics of conversations [Power 74, 77] imposed more deep thought about the inner organization of agent mentality, and new mental states were needed to capture agent's attitudes around motivation and flexibility along interactions: intentions, desires and expectations. The role of context was determinant, but the reason for that is behind the pragmatic nature of conversations. The first results were encouraging, but extended experimentation is still required.

3- From Descriptions to Systems and Back.

There is a number of views on how to make sense out of our work in Artificial Intelligence. Some people say they want to understand how the human mind works (strong thesis), and so they lead their research along this line. Some other people defend that they don't need to have that pretension and just want to provide new and clever ways to solve problems (soft thesis). There is some hope that these results can be extrapolated and contribute to a more general theory of intelligence (or human mind, or human behaviour).

Consider figure 3. The cube on the left represents the world and the sign on the right represents the observer, i.e. the agent that perceives and acts in the world. This agent has several different views on the world, which are represented by the squares on the middle.

What we want to do (and we place ourselves as the observer on the bottom) is to understand what is the structure of an agent and how does this structure (in terms of architecture and mechanisms) allow the agent to have his several views on the world and to reason and decide things in such a way that he acts on the world in a manner we consider intelligent (or, at least, interesting).

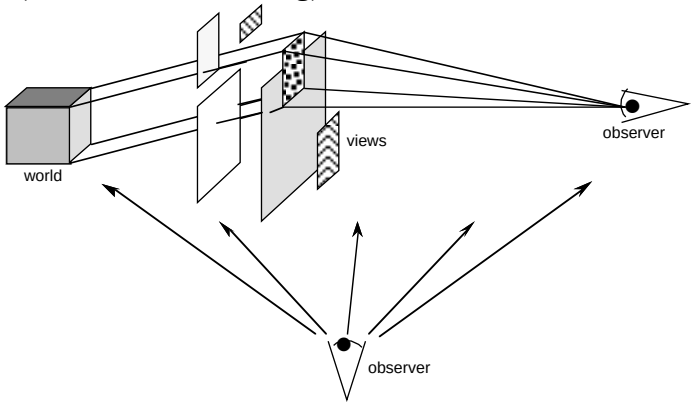

Figure 3: The relation between an observer and the world.

Unfortunately, there aren't many ways for us to understand how this agent is constituted, and how it operates. The usual way to do this is to place ourselves as the agent and try to analyze how we build those views on the world. This process is called introspection. We can then observe what happens as the observer changes from one view to another, and what happens when the observer changes position (and therefore change his set of views).

From these observations, we hope we can get enough information to be able to choose one formalism for representation. In the common kind of Artificial Intelligence empirical work, after this choice, we design a (computational) system that can exhibit the behaviours we were trying to study and produce.

The engineering process we have just described can be seen as a voyage along an axis, like the one depicted in figure 4, from higher to lower level representations.

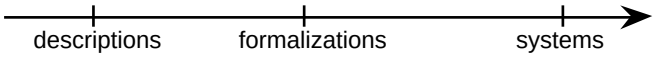

Figure 4: From descriptions to systems.

But this view is a reductive and simplistic one. For not only we have other axes involved, but also the voyage is not just from left to right. Let's analyze this in detail.

Typically, the first thing we have to do is what we will call 'capture' the world. We will not use the usual term perceive, since we believe that a little more takes place when an observer looks at a situation for the first time. If we had to use an equation, we would write: 'to capture = to perceive + to understand.' In the perception task, there is the notion of relevance involved. We perceive what is relevant for us to perceive. This relevance has to do with the fulfillment of our motivational states (desires, intentions, or obligations). Also, we make sense out of the world by getting from reality the information we need to get and inserting it into the right holes in a schema or frame we have in our minds (model). This frame is build with the help of years of experience and education, including formal education. So, even in this early stage, we see that there are already interiorization processes that have to do with (for instance) the formal machinery we have at our disposal.

Our next step is to describe the world. This description is done using an informal language, and it is meant to be a high level specification of the problem we are addressing. It is usual to discover errors (or refinements) in previous stages, in this case, perception errors. These are not necessarily wrong observations, they could come from lack of information, or failure to get understanding on the collected data. Although informal, the language we adopt must be as unambiguous as possible, since ambiguity would raise problems when we passed to the following stages of the development process. We shouldn't forget that the informal description (or descriptions) we provide may allow us to

clearly identify the concepts involved in our problem, the relations among them and how they will evolve with time.

The next step is the construction of formalizations of the situation. Since there is a number of well-established formalisms, some people tend to provide informal descriptions that are already especially suited to fit them. This way, they ease up the translation from the descriptions to the chosen formalisms. But this is an awkward form of posing the question. We don't choose problems or situations to suit the formalisms we like. In point of fact, what we do is to consider and describe situations and then see which is the formalism (if there is one, or else we have to define one) that suits the description the best. On the other hand, in order to avoid ambiguity, we must provide successive descriptions that are closer and closer to the formalism we have chosen. Nevertheless, when we choose a formalism before having a description, there is the danger of loosing interesting information at a very early stage. It is the descriptions that have to be increasingly formal, loosing in information (through abstraction) and gaining in formalism as the successive languages loose in expressive power and gain in tractability and correctness. Also, the translations must be well understood. Unfortunately, they are difficult to analyze formally, but we must have the notion of what is being done, so that we are able to work backwards and re-engineer if necessary. So, another evaluation and refinement process is needed for every translation we make. In some cases the description wouldn't have been correct, in others the problem is with the translation, and, finally in other ones, the trouble is with the (more formal) language we have chosen.

When we have a formal specification, we can develop a system that implements it. The implementation of the system can be viewed as the final of a series of translations, starting with very high level languages and ending with programming languages, that can be less expressive, but have other favourable features, like computational efficiency, development environments, or analytic facilities (e.g. to see what happens inside the agents' head during the experiments).

It is common to conceive the whole journey as composed of only three steps: providing a description, formalizing, and building the system (e.g. MAD: modelling, analysis, and design). It can then be argued that it suffices to provide a correct algorithm to translate the specification into a system. The correction of the algorithm can be defined as the maintenance in the system of the properties that can be established in the formalization. The only problem that would remain is to provide adequate formalisms to represent the informal descriptions.

But several points have to be made. First of all, the need to make the translation back to previous languages (and abstraction levels). This is necessary both to debug and to evaluate the descriptions (and ultimately the working system). It isn't evident that an absolutely correct algorithm can be built, one that avoids the need to debugging. Think, for instance, of a first system that simulates a robot's arm functioning and a second system composed by a real arm and the software that operates it. The kind of control required by the second system is so different from the first that it is to be expected that some problems arise. How to debug the system, or to account for unexpected features in the system behaviour, if we cannot relate our observations to the specification we made?

On the other hand, we can have final systems that do not differ so much from the formal specifications. They could be symbol manipulating programs. In this case, where is the relation between the symbols in our system and the objects in the world? Why is this relation more present in the working system than it would be in the specification? The answer to these questions is: it depends on what happens in the designer's mind. But it is easy for him to draw those relations, because he has the global picture of the whole path from the initial description to the desired system. There is no less need for the journeys back and forth, since in these conditions it is easy to wrongly introduce features in a level that weren't in a previous one, for instance, solutions to unspecified problems.

4- Experiments in InsectWorld.

The soundness of theoretical ideas and results on the study of agent models is better evaluated by experimental settings. But, evaluation in empirical Artificial Intelligence is not so clear like in behavioural sciences, where it includes describing experiment designs, results and analyses. In AI, this is difficult because experiments often involve uncontrolled interactions of knowledge representations, inference methods, algorithms, and the user [Cohen et al 89]. And, yet more complex in DAI, where distributed behaviours are central.

After the experimentation carried out by [Caldas and Coelho 92], within the context of a particular problem domain, economic transactions (non normative behaviours based upon changes of the agents' mental states), we decided to build up an experiment schema where several lines of research around the best set of mental states, the agent architecture, and the types of interactions (e.g. exchanges, conversations, negotiations) among agents were integrated around normative behaviours in social organizations (several actions based upon relations of dependence, power, communion, and sharing of interests and objectives).

From the beginning we were not interested on performance evaluations, but on the adequacy of our agent models to particular tasks where normative reasoning is required. Namely, we were keen motivated to understand how implementations (programs) reflect accurately formal descriptions, and how translations are done back and forward. We shall describe our path in developing a collection of experiments, and use this description as an example of the methodological issues we want to point out.

In [Moniz 93a], it was proposed a prototype of a workbench for development and test of agent architectures. The main features of this workbench are (1) its generality, in the sense that it doesn't fix constraints on the agents' structure, (2) the possibility for the agents to have different and concurrent units, and (3) the easiness to build simulated environments where to place experiments, since the environment can be programmed as any other agent.

Following the work about intention and its relation to action [Antunes 93, Corrêa and Coelho 93], and taking into consideration the proposals of [Castelfranchi 90, 93] about power and social dependence, we tried to build up a collection of experiments aiming at (1) verifying the options adopted for the workbench, (2) exposing the current agent architectures to a new concept, such as power, in order to derive some conclusions about their adequacy, or otherwise some insights on how they could be improved, and (3) studying the concepts related to power and getting some experimental results to confront with Castelfranchi's theories.

From the very beginning, our perceptions on the problem were conditioned by a number of factors. First, we had already some formalisms in mind, and also a system and a language where we intended to develop the experiments. Like we said in the previous section, these pre-fixed ideas constrain the descriptions we make, causing them to be especially adapted to the formalism or programming language, and possibly loosing important information on account of that.

Second, we decided to explore a previous experiment further on, in which we had tried out a model of intentions achievement [Antunes et al 93]. This provided a simulated environment and some simple agents that we could expand, instead of beginning from scratch. Nevertheless, it should be noted that this was another heavy constraint, and it wasn't absolutely necessary, rather it was an option made to shorten only the programming phase.

Finally, the ideas we followed were inspired in Castelfranchi's descriptions (some of them quasi-formal, à la Cohen and Levesque), which of course reflected his own ideas on the subject of power and its relations to agency. Open as we want to be when designing an experiment, there can always be important features that are not only left out in our description, but also there is the danger of our original frame of mind not being able to support these features, and so they might never appear. It is because of problems like these that we introduce new concepts in existing schemas, like we tried (and are trying) to do with power. Let us pick up only the first experiment of that collection, to make clear our ideas.

Statement of Experiment 1.

Two ladybirds lay in an environment (12-by-13 square world) where they must consume resources in order to survive. There are two resources they must take: water and fruit. The water resource is dominated by ladybird A, who cannot directly access the other one. The ladybird B must deliver fruits to ladybird A in order to get water.

Informal Description of Experiment 1.

This experiment was built up to study how can power relations be interpreted and used. In this example, both agents are linked together: agent A has power over B, he can prevent B from getting water. On the other hand, A is dependent on B for getting fruits.

When agent A is hungry he starts to execute a plan to get fruits. The first stage of this plan consists in making a promise to agent B. This promise originates a new concept on agent A: he adds to his beliefs the responsibility of giving water to agent B when B gives him fruits. This is not mandatory for agent A, he has no rigid commitment to agent B. The promise is only carried out if the agent has water. When agent A makes the promise he doesn't know if he will have water to give to agent B. But on the other hand, if he does have water, he will stick to his promise. In figure 5 we have a very coarse grained view of a cognitive agent's structure.

The experiment evolves in an environment described as a square space, where fruits grow at a preset rate and water is sometimes available. Each square position can be occupied by only one agent, but it can hold simultaneously an agent and a fruit (or water).The agents can move from a square to any other adjacent one, in one of the cardinal directions. Other possible actions are to pick objects (fruits or water), to consume them (eat or drink) or give them away, or to engage in interactions with other agents.

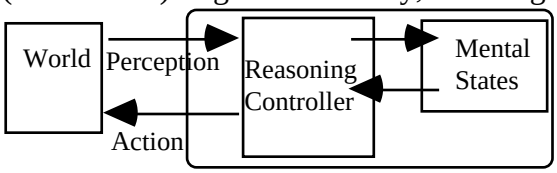

Figure 5: The agent's structure.

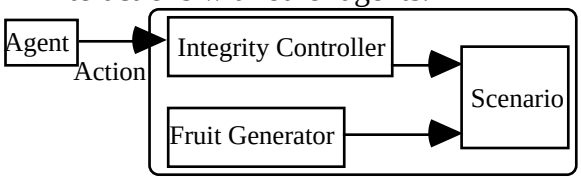

Figure 6: The world structure.

The world controls the agent's actions. For instance, if an agent tries to move to an occupied position, the world doesn't update the position of the agent. It's possible, in result of that action, that the agent's internal representations become different from his real position. This option gives us the ability to build imperfect agents, that are able to evolve without producing incongruences (moving to an occupied position) in the world.

In figure 6 we present the structure of the world for these experiments. The rules that determine what is possible to happen in the world are represented in the integrity controller. They determine if an action is valid or not. The fruit generator only makes a fruit appear when some situation occurs, for instance some time has passed, or some agent has eaten a fruit. There is no single agent that has direct control over the scenario. We can also see these rules as social norms that regulate the interactions among agents and the world (for instance they can represent the laws of physics).

Reasoning Strategies. Normative and Power Aspects.

The behaviour of agentA is constrained by a set of rules. In the interaction with agentB, agentA makes a promise of giving water in exchange for a fruit. AgentA is compelled by his construction to fulfill the promise (he adds to his beliefs the concept of promise).

The promise that agentB does is not restrictive to his behaviour, and at any time he can interrupt the plan he is executing and carry out other tasks. The promise is forgotten, or carried out later on. He hopes to be rewarded by giving the fruit to agentA.

In terms of power relations, both agents have the ability to prevent the other one from achieving his goals. It is possible for agentA not to give water to agentB, and it is also possible for agentB to delay the fulfillment of his promise.

According to [Castelfranchi 90], there are two fundamental questions to face when dealing with several agents: (1) the sociality problem (why does an autonomous agent enter into social interactions?), and (2) the adoption problem (how does an agent get his goal to become social, i.e. get it adopted by other agents?).

The answer to the first question is 'dependence.' We have avoided dealing with that by stating ourselves that the only way ladybirdB could get water was to get it from ladybirdA. In a more complex situation, there would be several ways of getting water, and ladybirdB would have to choose one of them, based on some selection criteria.

The answer to the second question is 'power.' Castelfranchi proposes several reasons for goal adoption, in order to avoid forced benevolence and subservience. He offers: (i) the agent wants to attain his own self-interested goal, (ii) he has altruistic reasons (affection, love, pity, even benevolence), (iii) he fears some situation, and adopts the goal because of threats or menaces, (iv) he has a commercial contract, and (v) he is given orders to do something (we could ask ourselves about what makes him obey those orders). In our present situation, ladybirdB adopts the other ladybird's goal for her own reasons, so that she can fulfill her own intention (to drink).This matches Castelfranchi's first postulate:

"Each new goal in an agent derives from his pre-existing goals (and assumptions)."

Anyway, something must be said about what makes one agent adopt the other agent's goal. In our case, it is agentB that adopt agentA's goal of having a fruit. Since what is involved here is the promise of a prize (the retribution of agentA, by giving water to agentB), it is assumed by agentA that the following goal-generating rule (planning rule) is valid in agentB's mind [Castelfranchi 90]:

BEL(Ag, IF(p, q)) & WANT(Ag, q) ===> WANT(Ag, p)

We can rewrite this rule as an inference rule. The following new formulation is closer to the formalisms we used in [Antunes et al 93] to study intention, which we will adopt ahead (see also [Antunes 93]).

{bel(Ag, plan(q, p)), int(Ag, q)} |– int(Ag, p)

In experiment 1 we adopted the above rule, reflecting the influencing power strategy of reward promising. We are presently conducting other experiments which make use of different influencing power strategies, for instance threat of sanctions, characterized by the following rule:

{bel(Ag, plan(q, p)), int(Ag, ¬ q)} |– int(Ag, ¬ p)

From the other agent's standpoint, we should think about why should he reciprocate. After ladybirdA has his fruit, he might as well not give water to ladybirdB. Castelfranchi points out the following reasons for reciprocation: (i) the agent has a final goal (moral imperative) to reciprocate, (ii) he wants to avoid guilt, (iii) he fears the society's punishment, (iv) he fears retaliation, and (v) he has a goal to retain the confidence for future exchanges.

In the modelled situation, we can say that our agent has a moral imperative to reciprocate, since he will always stick to his promise, if he has the conditions to fulfill it. This is clearly the simplest option we could have made, since any one of the others would require a sufficiently good understanding of the processes involved to allow for their modelling in terms of the formalism we choose.

For instance, to model a goal to retain confidence, and to establish a goal to reciprocate for that reason, we would have to understand the concept of confidence, and its role in the (future) establishment of contracts. It would imply the capacity to evaluate what would happen if he other agents don't trust our agent anymore.

We can model our agent's goal to reciprocate with the following rules. The first one states that Ag will keep his promise. The second one says that Ag makes his promise because he has agreed to do something if Ag' would do his part, and he believes that Ag' has done it. It would be possible to require less from Ag', for instance that Ag had the believe that Ag' would do his part of the deal.

{bel(Ag, promise(Ag', p))} |– int(Ag, p)

{bel(Ag, agreed(Ag', p, q)), bel(Ag, done(Ag', q))} |– bel(Ag, promise(Ag', p))

Let us see now how this type of formal knowledge can be used and explored in computational terms.

Formalization of the Situation.

From an engineering point of view, we can sketch our experiment by beginning with some sort of lower-level formal description. The formalism we use is a Prolog-like language, with two operators (intention and belief) for the case of cognitive agents. The formal descriptions have two complementary perspectives.

On the one hand, we have declarations, predicates about the states on the agents' minds. The cognitive agents have their mental states in terms of beliefs and intentions (only). Among the several possible beliefs, plans and actions play a relevant role. Actions are seen as 'basic actions' [Moya 90], i.e. actions the agents primitively know how to deal with. As for plans, they are just compositions of these primitive actions. A more appropriate model of plans is of course needed, but for our present experiment this simplification will suffice.

On the other hand, agents have procedures describing how they manipulate their mental states. In particular, cognitive agents have procedures to determine how they will fulfill their intentions (which are central to their behaviour), how they will execute their plans and actions, and how to add new propositions from the existing ones. For the agent that represents the world, procedures are seen as an integrity controller, assuring that the happenings and actions that occur make sense from the world physics standpoint.

In figure 7-a, we can see part of our formal description of the world environment.

Scenario

```
on(agentA,10,12)
on(agentB,5,3)
on(fruit,3,8)
on(fruit,8,2)
on(water,10,12)
```

Integrity Controller

```
verify-action <- read-message(message(Agent,Content)) & interpret(Content,Action) &
    execute(Action)

execute(move(Ag, Dir)) <- on(Ag,X,Y) & transform(X,Y,Dir,X1,Y1) & free(X1,Y1) &
    forget(on(Ag,X,Y)) & remember(on(Ag,X1,Y1))
execute(eat(Ag,fruit)) <- on(Ag,X,Y) & on(fruit,X,Y) & forget(on(fruit,X,Y))
…
```

Figure 7-a: Formalization of the world environment.

In the scenario part, we have some propositions setting the initial state of the world. It is assumed that no inconsistencies occur in this description. The integrity controller part describes how the actions are processed by the agent that represents the world. When it receives a message it interprets its content and executes the corresponding action. The only actions carried out are the ones that are explicitly foreseen at design-time. We show how some of these actions will be executed. Other actions are possible, namely to push an object, or to drink. By no means the description in figure 7-a is complete. It misses, for instance, the fruit generator, that enables fruits to be available during the simulation. It also misses other required mechanisms, like the one that takes care of message passing. We should also notice that there are a number of methods an agent has available to perform calculations, or to deal with its mental representations. As an example of the former we see in figure 7-a the method 'transform,' that given a position and a direction determines the new position in that direction. As examples of the latter type, we have the methods 'remember,' 'forget' and 'replace,' that establish the links between mental processes and mental states.

In figures 7-b and 7-c, we show the formalization of agents B and A.

Memory (Mental space: intentions, beliefs)

```
int(drink(water))
int(eat(fruit))
```

bel(plan(eat(fruit), search(fruit) & eat(fruit)))
bel(plan(drink(water), make-request(Agent, exchange(water,fruit)) & search(fruit) &
carry(fruit,Agent) & give(Agent,fruit) & wait(get(water)) & drink(water)))

Procedures (Reasoning capabilities)

fulfill(I) <- int(I) & bel(plan(I,P)) & execute-plan(P)
execute-plan(A1 & A2) <- execute(A1) & execute-plan(A2)
execute-plan(A1) <- execute(A1)
execute(search(fruit)) <- get-direction(fruit,D) & move(D) & false
execute(search(fruit)) <- on(agentB,X,Y) & on(fruit,X,Y) & remember(have(fruit))
…
execute(Action) <- send-message(Action, world)

Figure 7-b: Formalization of ladybird B.

Memory (Mental space: intentions, beliefs)

int(eat(fruit))

bel(hungry)
bel(have(water))

bel(plan(eat(fruit),make-request(Agent, fruit) & wait(get(fruit)) & eat(fruit) & give(Agent, water)))

Procedures (Reasoning capabilities)

fulfill(I) <- int(I) & bel(plan(I,P)) & execute-plan(P)
execute-plan(A1 & A2) <- execute(A1) & execute-plan(A2)
execute-plan(A1) <- execute(A1)
execute(eat(X)) <- send-message(eat(agentA,X), world) & forget(bel(hungry)) & forget(have(fruit))
…
execute(Action) <- send-message(Action, world)

Figure 7-c: Formalization of ladybird A.

In these formalizations, we try to capture the description of the experiment we outlined in figure 3. In ladybirdB's mental space we have declarations about its beliefs and intentions. In particular, the agent has two beliefs that represent plans to achieve some states: a state where the agent has drunk water and another where he has eaten a fruit. In the situation described in figure 7-b, the agent has exactly those two states as intentions to be fulfilled. If he had other intentions he wouldn't be able to carry out any plans to achieve them. This is due to the fact that we weren't concerned about the construction of plans. The agent's reasoning capabilities are centered around the fulfillment of its intentions. The method to do this is based on Castelfranchi's inference rule for promise modelling: {bel(Ag, plan(q, p)), int(Ag, q)} |– int(Ag, p). Nevertheless, in the method 'fulfill' we establish what it means to fulfill an intention, where in the inference rule we have just the birth of a new intention, and nothing is said (although it is implicit) about its fulfillment. On the other hand, in the method we don't fulfill the intention through the fulfillment of its sub-intentions, as it could be expected from the rule.

This is possible because the fulfillment of intentions is done through the execution of plans that are composed only of atomic actions, i.e. actions the agent knows how to deal with. These actions can have internal consequences (in terms of the agent's mental states) and external consequences. The latter are taken care by the environment. The former are dealt with by the agent's own 'execute' method. The plans we included are very simple, but a more elaborated executor of plans can be consulted in [Antunes et al 93], where we have both plans composed with different connectives and including actions that cannot be directly executed (and therefore leading to the formation of new (sub-)intentions and the execution of sub-plans).

The SSAE workbench and its Agent Definition Language (ADL).

The structure of the SSAE workbench [Moniz 93b] is composed of four sections: the agents, the libraries, the interpreter, and the implementation of parallelism. The agents-section contains the information concerning the agents we want to model. It also contains the information about the environment where the agents evolve. The libraries-part contains components (pieces) that can be used to build agents. These components implement essentially specific functions, like arithmetic operations or plan builders. The interpreter executes the agent definition language and supports some of the workbench primitives.

The parallelism controls the sequencing of the agents. This section is necessary due to the fact that the parallelism is simulated, in our present SSAE implementation.

The functions that allow the control of the simulation belong to the agents themselves, especially the one that represents the environment. In order to be able to define and test different agent architectures, the structure of an agent in SSAE must be as general as possible. The agent is seen as a parallel machine, capable of executing several functions simultaneously. These different functions (or activities) are implemented by the unit concept. Each of the units represents an independent part of the agent. The control part of the agent is in charge of sequencing the units for execution. Even in a true parallel implementation, we would still need some kind of control, to establish hierarchical levels between units. As for modules, each of them can be said to implement a function, given the satisfaction of some conditions. These actions can either consist of a change in the agent's internal states or have consequences in the world. Finally, an agent's methods are procedures or plan schemes to perform calculations, or verify internal situations.

Converting the Formalization to ADL.

The conversion of a logical description of an agent to the ADL language that will be interpreted by the workbench is a hard task. The problems arise due to the fact that we tend to describe the agent in a sequential manner, where declarative and procedural parts get confused. The agent has to be declarative in order to be executed in our workbench. It must have its state explicitly represented. This representation allows the workbench to suspend the agent execution in one point and restart from there later on (this limitation will be overdue in subsequent versions). It is not mandatory that the user observes this rule if he doesn't want his agent to be interrupted.

In figure 8 we present the representation of the agentB, with its initial internal states, in the workbench language ADL [Moniz 93a]. We must have the explicit representation of the plan the agent is executing. At each moment we can observe what he will do next. In the procedural section we can see that the carrying out of a plan is the execution of the first action of that plan and the elimination of that action from the executing plan. Each time the agent is activated, the method 'fulfill(I)' is called, and according to his current intentions and beliefs a new plan is started or an old one is reactivated. This simple mechanism allows the agent to change his mind if something goes wrong. For instance, if the agent wants to eat a fruit that no longer exists, he has no reason to proceed with the current plan.

Let us take now the conventional architecture of a SSAE workbench agent, as it is depicted in figure 8.

Control

```
if(int(eat(fruit)), activate(search-for-fruit),
  if(int(drink(water)), activate(search-for-water),
    activate(interact))))
```

Unit search-for-fruit

```
if(have(fruit), eat, search-fruit)
```

Unit search-for-water

```
if(have(water), drink, search-water)
```

Unit interact

```
interact
```

Module search-fruit

```
do(fulfill(eat(fruit)))
```

(Some modules are missing for the sake of conciseness)

Methods

```
fulfill(I) :- int(I), bel(executing(plan(I,P))), execute-plan(P).
fulfill(I) :- int(I), bel(plan(I,P)), execute-plan(P).

execute-plan(A1 & A2) :- execute(A1),
  replace(bel(executing(plan(I,_))),bel(executing(plan(I,A2)))).
execute-plan(A1) :- execute(A1), forget(bel(executing(plan(_,_)))).
execute(search(fruit)) :- get-direction(fruit,D), move(D), fail.
```

```
execute(search(fruit)) :- on(agentB,X,Y), on(fruit,X,Y), remember(have(fruit)).
…
execute(Action) :- send-message(Action,world).
```

Internal States

```
int(eat(fruit))
int(drink(water))

bel(plan(eat(fruit), search(fruit) & eat(fruit)))
bel(plan(drink(water), make(request(Agent,exchange(water,fruit))) &
  search(fruit) & carry(fruit,Agent) & give(Agent,fruit) & wait(get(water)) &
  drink(water)))
```

Figure 8: Ladybird B written in ADL.

In the control part of the agent, the unit which will be activated is selected, according to the agent's current intentions. Each of these units is responsible for one thread of the agent's behaviour. In figure 8, we have three units: 'search-for-fruit,' search-for-water,' and 'interact.' These units run in parallel, causing the agent's behaviour by calling modules. Here we have only one module, 'search-fruit,' which correspond to the fulfillment of an intention to eat fruit. The way to fulfill an intention is specified in the method 'fulfill.' Other methods state how to execute plans or actions. These methods can access the agent's internal states, namely intentions and beliefs.

The program in figure 8 shows our agent in the initial situation. The execution of the program will bring about different mental states, like the one shown in figure 9, where the agent (ladybirdB) is engaged in executing the plan to drink water, and his next action will be to give the fruit to agentA.

(Additional) Internal States

```
bel(executing(plan(drink(water), give(Agent,fruit) & wait(get(water)) &
  drink(water))))
```

Figure 9: Ladybird B is ready to give water to Ladybird A.

The Journey Backwards.

When we have reached this point, it seems to be the time to look back and evaluate whether the experiment we have built is well adequate to the problem we wanted to address. We should also check all the translations between formal descriptions and programs we have made, to see if they were properly done.

Let us imagine we are making a translation from one description D_i in a language L_i, to another language L_{i+1}. We don't want to loose the richness we have in D_i because of the features of L_{i+1}. On the other hand, we don't want to introduce characteristics on a description that are not present in a previous one, since that could alter the results of the experiments. We should also be careful not to introduce in D_{i+1} any features present in D_{i-j} ($1 \leq j < i$) that aren't in D_i. This would be rather simple an error to occur, since we usually have at all times a global view of what we want to describe and the system we ultimately want to develop.

In our example, we wanted to focus on a particular aspect of exercising power, the strategy of reward promising. After making observations of the working system we concluded that there was excessive control upon the agents. Namely, the fact that "ladybirdB wants to drink" is established once and for all (determinism), from the start, by the design. This negative result was not expected, although it could have been derived from the higher-level description. Also, in the study of the setting about exercising power (formation and fulfillment of a promise) and of the agent model a question was raised: "How could ladybirdA not fulfill its part of the deal, and how could it not keep its promise?" The conclusion is straightforward: it had no choice.

The modelling of the notion of choice was obligatory to fix again the experiment schema. As a matter of fact, this move could also improve the previous notion of intention.

Let us look back at our fist attempt to capture this notion, and at our proposals to model and program it. Taking our everyday knowledge about this problem, we decided to

simulate (and let the non-determinism be provided by the flow of the simulation) the functions which cause hunger and thirst as the source of motivation that leads to choice. In the following we present the revision of our first conceptualization and evaluate again the results obtained.

Formalizing the Source of Motivation.

The formalization of the notion of choice was accomplished by introducing two new beliefs in the agent's mental space, as we see in figure 10. The regulation over time of these now quantified concepts had to be provided, so that they could effectively (and autonomously from the agent's reasoning) model the motivations for choice.

So, we had to include other behaviours that are not controlled (or at least, not initiated) by the agent's reasoning scheme. These behaviours are dependent on other sources: the hunger of some agent must be constantly increased, regardlessly of any special reasons. In figure 11 we present a temporal logic description of this mechanism, as well as an analogous one for thirst. The first sentence says that in each instant of time hunger increases by one unit. The second sentence says that when the agent eats, hunger is decreased by a fixed quantity.

(Additional) Memory

bel(thirst(100))
bel(hunger(60))

Figure 10: Additional mental states for Ladybird B.

(Additional) Procedures (Bodily functions)

bel(hunger(Q)) -> Next(bel(hunger(Q+1)))
eat & bel(hunger(Q)) -> Next(bel(hunger(Q-K)))

bel(thirst (Q)) -> Next(bel(thirst (Q+1)))
drink & bel(thirst (Q)) -> Next(bel(thirst (Q-L)))

Figure 11: The mechanisms that represent bodily functions.

Only two aspects affect the agent's hunger (or thirst): the action of eating and the passing of time. The effects of the two aspects are obviously different. On the other hand, hunger and thirst have their own effects on the agent, namely as originators of intentions. This is represented in figure 12. The agent will have a new intention to drink water (through the execution of the method 'remember') when his thirst becomes greater than a certain quantity and he didn't have already that intention.

(More) Procedures (New intentions from bodily functions)

¬ int(drink(water)) & greater(thirst, T) -> execute(remember(int(drink(water))))

¬ int(eat(fruit)) & greater(hunger, H) -> execute(remember(int(eat(fruit))))

Figure 12: The birth of intentions.

A New Program for LadybirdB.

As a result of this new description and formalization, we have redone our previous program for agentB. In figure 13 we present the new ADL program for LadybirdB. Now, we have several units, that will run in parallel. This can be seen in the control part of the agent. Only the control section is new, and it is meant to replace the previous one, in figure 8. The units, modules and internal states sections are to be added to the figure 8 program.

(New) Control

```
if(int(eat(fruit)), activate(search-for-fruit),
  if(int(drink(water)), activate(search-for-water),
    activate(interact))))
activate(update-hunger)
activate(update-thirst)
```

(Additional) Unit update-hunger

```
update-hunger
```

(Additional) Unit update-thirst

```
update-thirst
```

(Additional) Module update-thirst

```
do(remember(int(drink(water)))) <- ¬ int(drink(water)) & greater(thirst,T)

do(replace(bel(thirst(X)), bel(thirst(X + 1)))) <- bel(thirst(X))
```

(Additional) Module update-hunger

```
do(remember(int(eat(fruit)))) <- ¬ int(eat(fruit)) & greater(hunger,H)

do(replace(bel(hunger(X)), bel(hunger(X + 1)))) <- bel(hunger(X))
```

(Some modules are missing for the sake of conciseness)

(Additional) Internal States

```
bel(thirst(100))
bel(hunger(60))
```

Figure 13: The new ADL program for Ladybird B.

	Agent A		Agent B		World	
Time	Mental Space	Received Messages	Mental Space	Received Messages	Scenario	Received Messages
t1	bel(have(water)) bel(hunger(40) …		bel(thirst(40)) bel(hunger(0)) …		on(agentA,8,8) on(agentB,5,8) on(fruit,6,6) on(fruit,4,3) on(water,8,8)	
t3	int(eat(fruit)) bel(have(water)) bel(hunger(42) …	(exchange (water,fruit), agentB)	int(drink(water)) bel(thirst(42)) bel(hunger(2)) …		on(agentA,8,8) on(agentB,5,8) on(fruit,6,6) on(fruit,4,3) on(water,8,8)	
t5	bel(promise (exchange (water,fruit), agentB)) int(eat(fruit)) bel(have(water)) bel(hunger(44) …		int(drink(water)) bel(thirst(44)) bel(hunger(4)) …	(ok (exchange (water, fruit)), agentA)	on(agentA,8,8) on(agentB,5,8) on(fruit,6,6) on(fruit,4,3) on(water,8,8)	
t7	bel(promise(…)) int(eat(fruit)) bel(have(water)) bel(hunger(46) …		bel(executing(plan(…))) int(drink(water)) bel(thirst(46)) bel(hunger(6)) …		on(agentA,8,8) on(agentB,5,7) on(fruit,6,6) on(fruit,4,3) on(water,8,8)	(move(up), agentB)
…	…		…		…	
t11	bel(promise(…)) int(eat(fruit)) bel(have(water)) bel(hunger(50) …		bel(have(fruit)) bel(executing(plan(…))) int(drink(water)) bel(thirst(50)) bel(hunger(10)) …		on(agentA,8,8) on(agentB,6,6) on(fruit,6,6) on(fruit,4,3) on(water,8,8)	(carry(fruit, down), agentB)
…	…		…		…	
t16	bel(promise(…)) int(eat(fruit)) bel(have(water)) bel(hunger(55) …	(take(fruit), agentB)	bel(executing(plan(…))) int(drink(water)) bel(thirst(55)) bel(hunger(15)) …		on(agentA,8,8) on(agentB,8,7) on(fruit,8,8) on(fruit,4,3) on(fruit,11,2) on(water,8,8) …	
…	…		…		…	
t20	bel(have(water)) bel(hunger(3) …		bel(executing(plan(…))) int(drink(water)) bel(thirst(59)) bel(hunger(19)) …	(take(water), agentA)	on(agentA,8,8) on(agentB,8,7) on(fruit,4,3) on(fruit,11,2) on(water,8,8) …	

Figure 14: Evolution of the experiment.

In figure 14 we present the evolution of the mental space of the agents and their interactions along the experiment. In time instant t1, we see the initial situation of our simulation. We still don't have any intentions in our agents. In instant t3 we can see them,

born due to the increase of the values for hunger and thirst. This increase continuously happens during the whole simulation, as a result of the execution of the parallel units that simulate the correspondent functions.

We can also see in t3 that agentA has received a message from agentB, proposing the exchange of a fruit for water. AgentB is executing his plan to get water, according to his beliefs. In instant t5 the deal is closed, when agentA agrees to the exchange. So, in instant t7, the agentB starts to execute his plan to get a fruit, moving towards it. To that purpose, he sends a message to the world, asking to be moved up one position.

After some time, the simulation reaches a state (instant t11) in which agentB is in square (6, 6), where the fruit he was searching is. AgentB starts to carry the fruit to the position where agentA is. In instant t16 agentB gives the fruit to agentA, who reciprocates in instant t20, giving water to agentB. These actions are executed by passing the corresponding messages from one agent to the other. We should notice that in instant t20, agentA doesn't have neither his belief about the promise to agentB, nor his intention to eat, anymore. Also, agentA's hunger has been reduced, due to his eating the fruit agentB has just given him.

The methodology suggested by [Cohen et al 89] is not sufficient to think about the architecture of an agent because there is something more to catch than its behaviour and the environment setting. Along this section we tried to discuss the agent design rationale by moving around the whole design space and by adopting several perspectives and tools at different levels [Newell 81]. Our alternative methodology has a broader scope than Cohen's one because it includes and covers the mind navigation maps of the designer keeping up the description consistency among translation levels. However, we need further work on the formal and system layers, not only to find the best tools, but also to isolate their most suitable primitives. Without that, the rigour is always absent along the translation process.

5- Conclusions.

"The ultimate value of experimentation is to constrain or otherwise inform the designer of a system that solves interesting problems."
[Hanks et al 93]

In this paper we discuss a (the) method for the validation of theories by working in different levels of description, either on the formal side or on the computational side, and by having the suitable means to cross from one side to the other and back, and also back to informal descriptions.

It is not known how we may relate the mentality (set of mental states) of an agent to a specific task, how mental states are properly defined (no consensus yet in DAI) and the only way out is the promise of controlled experimentation. But, the elucidation of the relationships between sets of properties of the agent architecture, environment and behaviour is not sufficient, because the languages behind have less expressive power than any formal language. We need tools for automatically visualizing the agents and for relating that to our own ability to mentally simulate the behaviour of agents, and also to evaluate the observations taken over all aspects of one's research. So, the main contribution of the present paper was to make clear that formalists and engineers must work together around models, algorithms and systems [Cohen 91] in order to increase the accuracy of DAI research.

Acknowledgements.

The research reported in this paper was partially carried out within the portuguese Project no. STRDA/C/TIT/86/92-JURAD, financially supported by the Program STRIDE, FEDER and JNICT.

References.

[Antunes 93] Luis Antunes, Intenção e Acção, MSc. Thesis, IST/UTL, Lisboa, July 1993.
[Antunes et al 93] Luis Antunes, Luis Moniz and Helder Coelho, Experiments with Intention-based Behaviour, Working Report, November 1993.
[Bratman 84] Michael E. Bratman, Two Faces of Intention, The Philosophical Review, no. 93, July 1984.
[Bratman 87] Michael E. Bratman, Intentions, Plans and Practical Reasoning, Harvard University Press, London, 1987.

[Bratman 90] Michael E. Bratman, What Is Intention?, in Intentions in Communication, Philip R. Cohen, Jerry Morgan and Martha E. Pollack (Eds.), MIT Press, 1990.
[Caldas and Coelho 92] José Caldas and Helder Coelho, Strategic Interaction in Oligopolistic Markets: Experimenting with Real and Artificial Agents, Proceedings of the Fourth European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Rome, July 29-31, 1992.
[Cascalho 93] José Cascalho, Diálogo entre Agentes, Graduation Report, IST/UTL, Lisboa, July 1993.
[Castelfranchi 90] C. Castelfranchi, Social Power. A point missed in Multi-Agent, DAI and HCI., in Decentralized AI - Proceedings of MAAMAW'90, Yves Demazeau and J.P. Müller (Eds.), Elsevier Science Publishers B. V., 1990.
[Castelfranchi 93] C. Castelfranchi, Commitments: from individual intentions to groups and organizations, in AAAI Workshop on Distributed Artificial Intelligence, 1993.
[Coelho et al 92] Helder Coelho, Graça Gaspar and Isabel Ramos, Experiments on Achieving Communication in Communities of Autonomous Agents, Proceedings of the IFIP 8.3 Working Conference on Decision Support Systems: Experiences and Expectactions, Fontainebleau, 1-3 July, 1992.
[Cohen 91] Paul R. Cohen, A Survey of the Eighth National Conference on Artificial Intelligence: Pulling Together or Pulling Appart?, AI Magazine, vol. 12, no. 1, Spring 1991.
[Cohen et al 89] Paul R. Cohen, Michael L. Greenberg, David M. Hart and Adele E. Howe, Trial by Fire: Understanding the Design Requirements for Agents in Complex Environments, AI Magazine, Fall 1989.
[Cohen and Levesque 87] Philip R. Cohen and Hector J. Levesque, Intention = Choice + Commitment, Proceedings of AAAI'87, 1987.
[Cohen and Levesque 90a] Philip R. Cohen and Hector J. Levesque, Persistence, Intention, and Commitment, in Intentions in Communication, Philip R. Cohen, Jerry Morgan and Martha E. Pollack (Eds.), MIT Press, 1990.
[Cohen and Levesque 90b] Philip R. Cohen and Hector J. Levesque, Rational Interaction as the Basis for Communication, in Intentions in Communication, Philip R. Cohen, Jerry Morgan and Martha E. Pollack (Eds.), MIT Press, 1990.
[Corrêa and Coelho 93] Milton Corrêa and Helder Coelho, Around the Architectural Agent Approach to Model Conversations, Proceedings of the Fifth European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Neuchâtel (Switzerland), 24-27 August, 1993.
[Costa et al 92] Ernesto Costa, Helder Coelho and Graça Gaspar, A Formal Approach to Intelligent Learning Environments: A First Tentative, Proceedings of the NATO Advanced Research Workshop on Student Modelling, Montreal (Canadá), May, 1991, and Springer-Verlag, 1992.
[Demazeau and Müller 90] Demazeau, Y. and Müller, J.-P., Decentralized AI, Proceedings of the First European Workshop on Modelling Autonomous Agents in a Multi-Agent Worl, Cambridge, England, August 16-18, 1969, North-Holland, 1990.
[Devlin 91] Keith Devlin, Login and Information, Cambridge University Press, 1991.
[Ferguson 92] Innes A. Ferguson, TouringMachines: Autonomous Agents with Attitudes, Technical Report, no. 250,Computer Laboratory University of Cambridge, Apr 1992.
[Gaspar 91] Graça Gaspar, Communication and Belief Changes in a Society of Agents: Towards a Formal Model of an Autonomous Agent, Proceedings of the 2nd European Workshop on Modelling Autonomous Agents in Multi-Agent Worlds, August 1990, and in Decentralized AI 2, North-Holland, 1991.
[Goldman and Lang 93] Robert P. Goldman and R. Raymond Lang, Intentions in Time, Tech. Rep. TUTR 93-101, Computer Science Dept. and CAACS, Tulane University, January 1993.
[Hanks et al 93] Hanks, S., Pollack, M. E. and Cohen, P. R., Benchmarks, testbeds, controlled experimentation and the design of agent architectures, AI Magazine, volume 14, no. 4, Winter, 1993.
[Hewitt 77] Hewitt, C., Viewing Control as Patterns of Passing Messages, Artificial Intelligence, volume 8, 1977.
[Hewitt 85] Hewitt, C., The Challenge of Open Systems, Byte, April, 1985.
[Jones 93] Jones, A. J. I., Practical Reasoning, California-style: some remarks on Shoham's Agent-Oriented Programming (AOP), Working Report, Dept. of Philosophy, University of Oslo, 1993.
[Junglaus et al 91] Junglaus, R., Saake, G., Hartmann, T. and Sernadas, C., Object-oriented specification of information systems: The Troll Language, TEch. Universität Braunschweig, 1991.
[Konolige and Pollack 93] Kurt Konolige and Martha Pollack, A Representationalist Theory of Intention, Proceedings of IJCAI, 1993.
[Minsky 75] Marvin Minsky, A Framework for Representing Knowloedge, on The Psychology of Computer Vision, P. Winston (ed.), McGraw-Hill, 1975.
[Moya 90] Carlos J. Moya, The Philosophy of Action - an introduction, Polity Press and Basil Blackwell Inc., 1990.
[Moniz 93a] Luis Moniz, SSAA: Sistema para Simulação de Agentes e Ambientes, MSc. Thesis, IST/UTL, Lisboa, June 1993.
[Moniz 93b] Luis Moniz, SA&A: Uma Bancada para Simulação de Ambientes e Agentes Heterogéneos, Anais do X Simpósio Brasileiro de Inteligência Artificial, Porto Alegre, Brasil, Outubro 1993.
[Newell 81] Allen Newell, The Knowledge Level, AI Magazine, vol. 2, no. 2, Summer 1981.
[Pollack 91] Martha E. Pollack, Overloading Intentions for Efficient Practical Reasoning, Noûs, vol. 25, no. 4, 1991.
[Pollack 93] Martha Pollack, Personnal Communication, 1993.
[Pörn 70] Ingmar Pörn, The Logic of Power, Basil Blackwell, 1970.
[Power 74] Power, R., A Computational Model of Conversation, Ph. D. Thesis, Department of Machine Intelligence, University of Edinburgh, 1974.
[Power 79] Power, R., The Organization of Purposeful Dialogues, Linguistics 17, 1979.
[Sernadas et al 87] Cristina Sernadas, Helder Coelho and Graça Gaspar, Communicating Knowledge Systems: Parts I and II—Big talk among small systems, Applied Artificial Intelligence, vol. 1, no. 3 and 4, 1987.
[Sernadas et al 91] Sernadas, C., Resende, P., Gouveia, P. and Sernadas, A., In-the-large object-oriented design of information systems, in Object-Oriented Approach in Information Systems, F. van Assche, B. Moulin, C. Rolland (eds.), Elsevier Science Pub. (North-Holland), 1991.
[Shoham 90] Shoham, Y., Agent-Oriented Programming, Technical Report, STAN-CS-90-1335, Computer Science Dept., Stanford University, 1990.
[Shoham 91] Shoham, Y. Implementing the intentional stance, in R. Cummings and J. Pollock (eds.), Philosophy and AI: Essays at the Interface, The MIT Press, 1991.
[Shoham 93] Shoham, Y., Agent-Oriented Programming, Artificial Intelligence Journal, volume 60, Number 1, March, 1993.
[Sloman 93] Aaron Sloman, Prospects for AI as the General Science of Intelligence, in Prospects for Artificial Intelligence - Proceedings of AISB'93, Birmingham, IOS Press, 1993.
[Sloman 94] Sloman, A. et al., Explorations in Design Space, Proceedings of the European Conference Aartificial Intelligence, Amsterdam (Holland), August 8-12, 1994.
[Steels 90] Luc Steels, Cooperation Between Distributed Agents Through Self-Organization, in Decentralized AI, Yves Demazeau & Jean-Pierre Müller, Elsevier Science Publishers B.V., 1990, pp. 175-196.
[Werner 91] Eric Werner, A Unified View of Information, Intention and Ability, in Decentralized AI 2 - Proceedings of MAAMAW'91, Yves Demazeau and Jean-Pierre Müller (Eds.), MAAMAW, Elsevier Science Publishers B. V., 1991.