1

Formal Methods and Agent-Based Systems

Michael Luck and Mark d'Inverno

1.1 Introduction

As has been discussed elsewhere [17], much agent-related work has tended to focus on either the development of practical applications, or the development of sophisticated logics for reasoning about agents. Our own view is that work on formal models of agent-based systems is valuable inasmuch as they contribute to a fundamental goal of computing to build real agent systems. This is not to trivialise or denigrate the effort of formal approaches, but to direct them towards integration in a broader research programme. In an ongoing project that has been running for several years, we have sought to do exactly that through the development of a formal framework, known as SMART, that provides a conceptual infrastructure for the analysis and modelling of agents and multi-agent systems on the one hand, and enables implemented and deployed systems to be evaluated and compared on the other. In this paper, we describe our research programme, review its achievements to date, and suggest directions for the future.

In particular, we consider the role of autonomy in interaction. *Autonomy* is independence; it is a state that does not rely on any external entity for purposeful existence. In this paper, we use our existing agent framework to address the issues that arise in a consideration of autonomous interaction. We begin by considering several problems that are prevalent in existing models of interaction, and which must be addressed in attempting to construct a model of autonomous interaction. Then we introduce a previously developed agent framework on which the remainder of the paper is based. The next sections describe and specify an autonomous social agent that acts in an environment, the way in which it generates its goals, and finally how it interacts with others in its environment. We discuss how this can be viewed as a process of discovery, and what such a view usefully brings to the problem.

1.1.1 Theory and Practice

Though the fragmentation into theoretical and practical aspects has been noted, and several efforts made in attempting to address this fragmentation in related areas of agent-oriented systems by, for example, [14], [22], and [32], much remains to be done in bringing together the two strands of work.

This section draws on Luck's outline [17] of the ways in which some progress has been made with BDI agents, a well-known and effective agent architecture. Rao, in particular, has attempted to unite BDI theory and practice in two ways. First, he provided an abstract agent architecture that serves as an idealization of an implemented system and as a means for investigating theoretical properties [28]. Second, he took an alternative approach by starting with an implemented system and then formalizing the operational semantics in an agent language, AgentSpeak(L), which can be viewed as an abstraction of the implemented system, and which allows agent programs to be written and interpreted [27].

In contrast to this approach, some work aims at constructing directly executable formal models. For example, Fisher's work on Concurrent MetateM [11] has attempted to use temporal logic to represent individual agent behaviours where the representations can either be executed directly, verified with respect to a logical requirement, or transformed into a more refined representation. Further work aims to use this to produce a full development framework from a single high-level agent to a cooperating multi-agent system. In a similar vein, [25] aims to address the gap between specification and implementation of agent architectures by viewing an agent as a multi-context system in which each architectural component is represented as a separate unit, an encapsulated set of axioms, and an associated deductive mechanism whose interrelationships are specified using bridge rules. Since theorem-provers already exist for multi-context systems, agents specified in this way can also be directly executed.

As yet, the body of work aimed at bridging the gap between theory and practice is small. Fortunately, though, there seems to be a general recognition that one of the key roles of theoretical and practical work is to inform the other [8], and while this is made difficult by the almost breakneck pace of progress in the agent field, that recognition bodes well for the future. Some skeptics remain, however, such as Nwana, who followed Russell in warning against *premature mathematization*, and the danger that lies in wait for agent research [4].

1.1.2 General Approach

As stated above, we view our enterprise as that of building programs. In order to do so, however, we need to consider issues at different points along what we call the *agent development line*, identifying the various foci of research in agent-based systems in support of final deployment, as shown in Figure 1.1. To date, our work has concentrated on the first three of the stages identified.

- We have provided a formal agent framework within which we can explore some fundamental questions relating to agent architectures, configurations of multi-agent systems, inter-agent relationships, and so on, independent of any particular model. The framework continues to be extended to cover a broader range of issues, and to provide a more complete and coherent conceptual infrastructure.

- In contrast to starting with an abstract framework and refining it down to particular system implementations, we have also attempted to start with specific deployed systems and provide formal analyses of them. In this way, we seek to move backwards to link the system specifications to the conceptual formal framework, and also to provide a means of comparing and evaluating competing agent systems.
- The third strand aims to investigate the process of moving from the abstract to the concrete, through the construction of agent development methodology, an area that has begun to receive increasing attention. In this way, we hope to marry the value of formal analysis with the imperative of systems development in a coherent fashion, leading naturally to the final stage of the development line, to *agent deployment*.

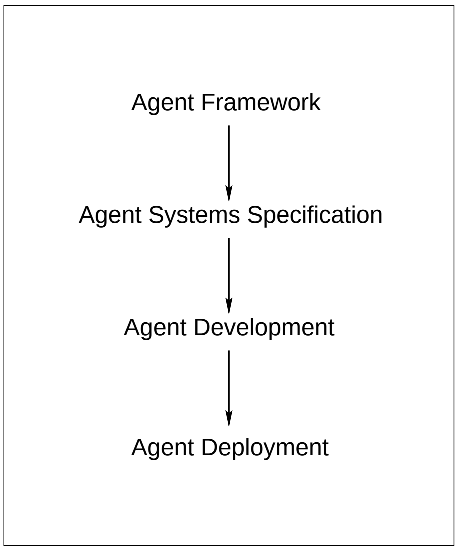

Fig. 1.1. The Agent Development Line.

This paper can be seen as an extension of the work contained in [21], which describes the results of the research programme in providing a foundation for the exploration of more advanced issues in agent-based systems. That work introduced requirements for formal frameworks, and showed how our agent framework satisfied those requirements in relation to, for example, some initial inter-agent relationships, and their application to the Contract Net Protocol. In this paper, we build on that work, showing further levels of analysis of agent relationships, and also describe further work on formal agent specification.

In what follows, we use the Z specification language [31] , for reasons of accessibility, clarity and existing use in software development. The arguments are well-rehearsed and can be found in many of the references given at the end of the paper.

Here, we present a brief introduction to the language but more details can be found in an array of text books such as [31]. The specification in this paper is not intended to be complete, nor to provide the most coherent exposition of a particular piece of work, but to show how a broad research programme in support of the aims above is progressing. Details of the different threads of work may be found in the references in each of the relevant sections. In particular, this paper is concerned with the design of *autonomous* agents: what it means for an agent to be autonomous and what that entails for any adequate model of interaction between such agents. Complex environments admit an inherent uncertainty that must be considered if we are to cope with more than just toy problems. In such uncertain environments, an agent must be autonomous; an agent cannot know in advance the exact effects of its or others' actions. This is of paramount importance, and an agent must therefore be designed with a flexibility that enables it to cope with this uncertainty by evaluating it and responding to it in adequate ways.

1.1.3 Introduction to Z

The formal specification language, Z, is based on set theory and first order predicate calculus. It extends the use of these languages by allowing an additional mathematical type known as the *schema type*. Z schemas have two parts: the upper declarative part, which declares variables and their types, and the lower predicate part, which relates and constrains those variables. The type of any schema can be considered as the Cartesian product of the types of each of its variables, without any notion of order, but constrained by the schema's predicates.

It is therefore appropriate to liken the semantics of a schema to that of Cartesian products. For example, suppose we define a schema as follows.

Pair
$first : \mathbb{N}$
$second : \mathbb{N}$

This is very similar to the following Cartesian product type.

$$Pair == \mathbb{N} \times \mathbb{N}$$

The difference between these forms is that there is no notion of order in the variables of the schema type. In addition, a schema may have a predicate part that can be used to constrain the state variables. Thus, we can state that the variable, *first*, can never be greater than *second*.

Pair
$first : \mathbb{N}$
$second : \mathbb{N}$

$first \leq second$

Modularity is facilitated in Z by allowing schemas to be included within other schemas. We can select a state variable, *var*, of a schema, *schema*, by writing *schema*.*var*. For example, it should be clear that *Pair*.*first* refers to the variable *first* in the schema *Pair*.

Now, operations in a state-based specification language are defined in terms of *changes to the state*. Specifically, an operation relates variables of the state after the operation (denoted by dashed variables) to the value of the variables before the operation (denoted by undashed variables). Operations may also have inputs (denoted by variables with question marks), outputs (exclamation marks) and preconditions. In the *GettingCloser* schema below, there is an operation with an input variable, *new*?; if *new*? lies between the variables *first* and *second*, then the value of *first* is replaced with the value of *new*?. The value of *second* does not change, and the output *old*! is equal to the value of the variable *first* as it was before the operation occurs. The $\Delta Pair$ symbol, is an abbreviation for $Pair \wedge Pair'$ and, as such, includes in the operation schema all the variables and predicates of the state of *Pair* before and after the operation.

GettingCloser

$new? : \mathbb{N}$
$\Delta Pair$
$old! : \mathbb{N}$

$first \leq new?$
$new? \leq second$
$first' = new?$
$second' = second$
$old! = first$

To introduce a type in Z, where we wish to abstract away from the actual content of elements of the type, we use the notion of a *given set*. For example, we write $[NODE]$ to represent the set of all nodes. If we wish to state that a variable takes on some set of values or an ordered pair of values we write $x : \mathbb{P}\,NODE$ and $x : NODE \times NODE$, respectively. A *relation* type expresses some relationship between two existing types, known as the *source* and *target* types. The type of a relation with source X and target Y is $\mathbb{P}(X \times Y)$. A relation is therefore a set of ordered pairs. When no element from the source type can be related to two or more elements from the target type, the relation is a *function*. A *total* function ($\rightarrow$) is one for which every element in the source set is related, while a *partial* function ($\nrightarrow$) is one for which not every element in the source is related. A sequence (seq) is a special type of function where the domain is the contiguous set of numbers from 1 up to the number of elements in the sequence. For example, the first relation below defines a *function* between nodes, while the second defines a *sequence* of nodes.

$Rel1 = \{(n1, n2), (n2, n3), (n3, n2)\}$
$Rel2 = \{(2, n3), (3, n2), (1, n4)\}$

In Z, a sequence is more usually written as $\langle n4, n3, n2 \rangle$. The *domain* (dom) of a relation or function comprises those elements in the source set that are related, and the *range* (ran) comprises those elements in the target set that are related. In the examples above, $\operatorname{dom} Rel1 = \{n1, n2, n3\}$, $\operatorname{ran} Rel1 = \{n2, n3\}$, $\operatorname{dom} Rel2 = \{1, 2, 3\}$ and $\operatorname{ran} Rel2 = \{n2, n3, n4\}$. Sets of elements can be defined using set comprehension. For example, the following expression denotes the set of squares of natural numbers greater than 10 : $\{x : \mathbb{N} \mid x > 10 \bullet x * x\}$.

For a more complete treatment of the Z language, the interested reader is referred to one of the numerous texts, such as [31].

1.2 Autonomous Interaction

Autonomy allows the design of agents to be flexible enough to function effectively and efficiently in a sophisticated world [5]. Typically, real autonomy has been neglected in most research. We hear of benevolent, altruistic, trusting, sympathetic or cooperative agents, yet a truly autonomous agent will behave only in a selfish way. Cooperation, for example, should occur only as a consequence of an agent's selfish motivations (which might of course include motivations relating to social acceptance that would drive what appears at face value to be "social" or "altruistic" behaviour). Autonomy allows for no artificially imposed rules of behaviour; all behaviour must be a consequence of the understanding and processing capabilities of that agent. Modelling this fundamental notion of selfish behaviour and the generation of goals by such a selfish autonomous agent is of vital importance in the design of autonomous agents.

In multi-agent systems, the interactions between agents are the basis for usefully exploiting the capabilities of others. However, such a pragmatic approach has not been the concern of many researchers who instead often focus on small areas of interaction and communication, and in particular on specialised forms of intention recognition and interpretation.

In many existing models of interaction, agents are not autonomous. In considering these models, we can identify problem-issues in autonomous interaction. Our intention is simply to show why these models are not adequate for autonomous interaction, and so isolate problems which contribute to the non-autonomous nature of these models.

Pre-determined Agenda Problem-solving can be considered to be the task of finding actions that achieve current goals. Typically, goals have been presented to systems without regard to the problem-solving agent so that the process is divorced from the reality of an agent in the world. This is inadequate for models of autonomy which require an understanding of how such goals are generated and adopted. Surprisingly, however, this is an issue which has received very little attention with only a few notable exceptions (e.g. [24]).

Benevolence In traditional models of goal adoption, goals are broadcast by one agent, and adopted by other agents according to their own relevant competence [30]. This assumes that agents are already designed with common or non-conflicting goals that facilitate the possibility of helping each other satisfy additional goals. Negotiation as to how these additional goals are satisfied typically takes the form of mere goal-node allocation. Thus an agent simply has to communicate its goal to another agent for cooperation in the form of joint planning to ensue. The concept of benevolence — that agents will cooperate with other agents whenever and wherever possible — has no place in modelling autonomous agents [7, 12]. Cooperation will occur between two parties only when it is considered advantageous to each party to do so. Autonomous agents are thus selfish agents. A goal (whether traditionally viewed as 'selfish' or 'altruistic') will always be adopted so as to satisfy a 'selfish' motivation.

Guaranteed Effects Speech Act Theory (SAT) [3, 29] underlies much existing work in AI [6], typically because as Appelt points out, speech acts are categorizable and can be modelled as action operators in a planning environment [2]. However, this work admits a serious flaw. Although the preconditions of these operators are formulated in terms of the understanding of the planning agent, the post-conditions or effects of these operators do not update the understanding of the planning agent, but of the agent at whom the action is directed [1]. No agent can ever actually *know* with any certainty anything about the effects of an action, whether communicative or otherwise. It is only through an understanding of the *target* agent and through observing the future behaviour of that agent, that the agent can discover the actual effects of the interaction. This uncertainty is inherent in communication between autonomous agents and must be a feature of any model of interaction which hopes to reflect this reality.

Automatic Intention Recognition A related though distinct problem with using SAT in the design of *communication* models involves the notion that the meaning of an utterance is a function of the linguistic content of that utterance. SAT is unable (even when one tries to force rather undistinguished ad-hoc rules [15]) to model any kind of utterance where the linguistic content is not very close to the speaker's intended meaning. That is to say that the operators themselves are context independent, and information about how context affects the interpretation of utterances is not explicitly captured. Communication varies from utterances with a meaning identical to linguistic content through utterances which have a meaning opposite to the linguistic content to utterances where the meaning does not seem to be categorised at all by the linguistic content. In short, Speech Act Theory cannot lead to a model of autonomous interaction. It merely serves to describe a very limiting case of linguistic communication at a suitable level for planning operators. A more flexible account of how intention is recovered from a multitude of different utterances is required.

Multi-Agent Modelling This is also a related but more subtle problem. Much work has modelled communicative actions in terms of mutual beliefs about the operator and its known effects [26]. This proposes to show not only how certain mental states lead to speech actions, but how speech actions affect mental states. We argue that any account of autonomous interaction should only model the effects of an action upon the mental state of the agent initiating the interaction (or another single agent).

In summary, there are several important claims here: first, an agent cannot be truly autonomous if its goals are provided by external sources; second, an agent will only adopt a goal and thus engage in an interaction if it is to its advantage to do so; third, the effects of an interaction cannot be guaranteed; fourth, the intentions of others cannot always be recognised; fifth, an agent can only know about itself.

Note that the first claim requires goals to be generated from within. It is this internal goal generation that demands an explicit model of the motivations of the agent. The second claim requires a notion of advantage that can only be determined in relation to the motivations of the agent. The third and fourth claims demand that the uncertain nature of autonomous interaction be explicitly addressed. We argue that viewing autonomous interaction as motivated discovery provides us with a means for doing this. Finally, the fifth claim imposes constraints on the problem we are considering, and provides a strong justification for our concern with constructing a model of autonomous interaction from the perspective of an individual agent.

1.3 The Formal Agent Framework

We begin by briefly reviewing earlier work. In short, we propose a four-tiered hierarchy comprising *entities*, *objects*, *agents* and *autonomous agents* [18]. The basic idea underlying this hierarchy is that all components of the world are entities. Of these entities, some are objects, of which some, in turn, are agents and of these, some are autonomous agents, as shown in Figure 1.2.

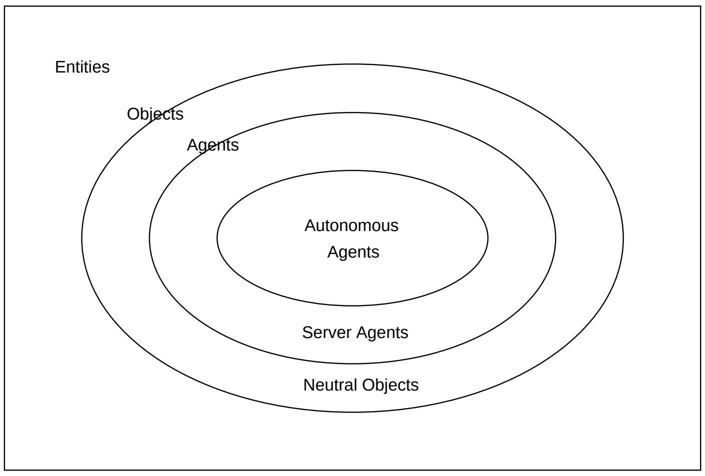

Fig. 1.2. The Entity Hierarchy.

Before it is possible to construct agent models it is necessary to define the building blocks or *primitives* from which these models are created. We start by defining three primitives: *attributes*, *actions* and *motivations*, which are used as the basis for development of the SMART agent framework. Formally, these primitives are specified as given sets which means that we say nothing about how they might be represented for any particular system. In addition, two secondary concepts, *goals* and *environments*, are specified in terms of attributes.

Attributes are simply features of the world, and are the only characteristics that are manifest. They need not be perceived by any particular entity, but must be potentially perceivable in an omniscient sense. This notion of a feature allows anything to be included such as, for example, the fact that a tree is green, or is in a park, or is twenty feet tall. An environment is then simply a set of attributes that describes *all* the features within that environment. Thus a type, *Environment*, is defined to be a (non-empty) set of attributes. The second primitive that needs defining is an *action*. Actions can change environments by adding or removing attributes. For example, the action of a robot, responsible for attaching tyres to cars in a factory, moving from one wheel to the next, will delete the attribute that the robot is at the first wheel and add the attribute that the agent is at the second. A goal defines a state of affairs that is desirable in some way. For example, a robot may have the goal of attaching a tyre to a car. A goal can therefore be very simply defined as a non-empty set of attributes that describes a state of affairs in the world. Lastly, *motivations* can be introduced. A motivation is any desire or preference that can lead to the generation and adoption of goals and that affects the outcome of the reasoning or behavioural task intended to satisfy those goals.

In summary we have the following definitions, from which we can construct an *entity* that has a non-empty set of attributes, and is therefore perceivable. For example, a goal is defined as a non-empty set of attributes.

$[Attribute, Action, Motivation]$

$Environment == \mathbb{P}_1\, Attribute$

$Goal == \mathbb{P}_1\, Attribute$

```
── Entity ──────────────────────────────
│ attributes : ℙ Attribute
│ capabilities : ℙ Action
│ goals : ℙ Goal
│ motivations : ℙ Motivation
├──────────────
│ attributes ≠ { }
└────────────────────────────────────────
```

1.3.1 Objects

Entities able to affect their environment through action lead to the concept of *objects*. An object in our model is an entity with *capabilities*. The *Object* schema below has a declarative part that simply includes the previously defined schema, *Entity*. The predicate part of the schema specifies that an object must have a non-empty set of actions as well as attributes. Objects are therefore defined by their ability in terms of their actions, and their characteristics in terms of their attributes.

$$\begin{array}{|l}
\multicolumn{1}{l}{\mathit{Object}} \\
\mathit{Entity} \\
\hline
\mathit{capabilities} \neq \{\,\} \\
\hline
\end{array}$$

Since an object has actions, these may be performed in certain environments that will be determined by the state of that environment. The behaviour of an object can therefore be modelled as a mapping from the environment to a set of actions that are a subset of its capabilities. This mapping is known as the *action-selection* function. The variable, *willdo*, specifies the next actions that the object will perform. Its value is found by applying the *objectactions* function to the current environment, in which it is situated.

$$\begin{array}{|l}
\multicolumn{1}{l}{\mathit{ObjectState}} \\
\mathit{Object} \\
\mathit{objectactions} : \mathit{Environment} \rightarrow \mathbb{P}\,\mathit{Action} \\
\mathit{willdo} : \mathbb{P}\,\mathit{Action} \\
\mathit{environment} : \mathit{Environment} \\
\hline
\mathit{willdo} = \mathit{objectactions}\ \mathit{environment} \\
\hline
\end{array}$$

An *interaction* with the environment occurs as a result of performing actions in it. Its effects on the environment are determined by applying the *effectinteract* function in the axiom definition below to the current environment and the actions taken.

$$\begin{array}{|l}
\mathit{effectinteract} : \mathit{Environment} \rightarrow \mathbb{P}\,\mathit{Action} \nrightarrow \mathit{Environment}
\end{array}$$

1.3.2 Agents and Autonomy

An agent is then just an object either that is useful to another agent where this usefulness is defined in terms of satisfying that agent's goals, or that exhibits independent purposeful behaviour. In other words, an agent is an object with an associated set of goals. This definition of agents relies on the existence of other agents that provide the goals that are adopted in order to instantiate an agent. In order to escape an infinite regress of goal adoption, we define *autonomous agents*, which are agents that generate their own goals from motivations.

$$\begin{array}{|l}
\multicolumn{1}{l}{\mathit{Agent}} \\
\mathit{Object} \\
\hline
\mathit{goals} \neq \{\,\} \\
\hline
\end{array}$$

$$\begin{array}{|l}
\multicolumn{1}{l}{\textit{AutonomousAgent}} \\
\textit{Agent} \\
\hline
\textit{motivations} \neq \{\,\} \\
\hline
\end{array}$$

For completeness we also define neutral objects as those objects that are not agents, and server agents as those agents that are not autonomous.

$$\begin{array}{|l}
\multicolumn{1}{l}{\textit{NeutralObject}} \\
\textit{Object} \\
\hline
\textit{goals} = \{\} \\
\hline
\end{array}$$

$$\begin{array}{|l}
\multicolumn{1}{l}{\textit{ServerAgent}} \\
\textit{Object} \\
\hline
\textit{motivations} = \{\} \\
\hline
\end{array}$$

1.3.3 Autonomous Perception

An agent in an environment can perceive certain attributes subject to its capabilities and current state but, due to limited resources, may not be able to perceive all attributes. The action of an agent is based on a subset of attributes of the environment, the agent's *actual* percepts.

To distinguish between representations of mental models and representations of the *actual* environment, a type, *View*, is defined to be the perception of an environment by an agent. This has an equivalent type to that of *Environment*, but now physical and mental components of the same type can be distinguished.

$$\textit{View} == \mathbb{P}_1\, \textit{Attribute}$$

An autonomous agent has a (possibly empty) set of actions that enable it to perceive its world, which we call its *perceiving actions*. The set of percepts that an autonomous agent is potentially capable of perceiving is a function of the current environment, which includes the agent's situation and its perceiving actions. Since the agent is resource-bounded, it may not be able to perceive the entire set of attributes and selects a subset based on its current goals. For example, the distributed Multi-Agent Reasoning System (dMARS) [10], may have a set of events to process, where events correspond to environmental change. Each of these percepts (or events) is available to the agent but because of its limited resources it may only be able to process one event, and must make a selection based on its goals.

The perception capabilities of an agent are defined in the schema below, *AutonomousAgentPerception*, which includes the *AutonomousAgent* schema and refines it by introducing three variables. First, the set of perceiving actions is denoted by *perceivingactions*, a subset of the capabilities of an autonomous agent. The *canperceive* function determines the attributes that are potentially available to an agent through its perception capabilities. Notice that this function is applied to a physical environment (in which it is situated) and returns a mental environment. The second argument of this schema is constrained to be equal to *perceivingactions*. Finally, the function, *willperceive*, describes those attributes actually perceived by an agent. This function is always applied to the motivations and goals of the agent and in contrast to the previous function, takes a mental environment and returns another mental environment.

$$\begin{array}{|l} \multicolumn{1}{l}{\textit{AutonomousAgentPerception}} \\ \hline \textit{AutonomousAgent} \\ \textit{perceivingactions} : \mathbb{P}\,\textit{Action} \\ \textit{canperceive} : \textit{Environment} \rightarrow \mathbb{P}\,\textit{Action} \nrightarrow \textit{View} \\ \textit{willperceive} : \mathbb{P}\,\textit{Motivation} \rightarrow \mathbb{P}\,\textit{Goal} \rightarrow \textit{View} \rightarrow \textit{View} \\ \hline \textit{perceivingactions} \subseteq \textit{capabilities} \\ \forall\, e : \textit{Environment};\ \textit{as} : \mathbb{P}\,\textit{Action} \bullet \\ \qquad \textit{as} \in \operatorname{dom}(\textit{canperceive}\ e) \Rightarrow \textit{as} = \textit{perceivingactions} \\ \hline \end{array}$$

To specify the actions of an autonomous agent, the next schema includes the *AutonomousAgent* schema and then defines an action-selection function that is determined in relation to the motivations, goals, perceived environment and actual environment of the agent. This function gives the set of actions the agent will perform, in order to achieve some goal. The physical environment is important here as it determines the action that is actually performed by the agent as opposed to the action which is intended (or selected) by the agent. These will not be the same in all cases as the agent may, for example, have incomplete or incorrect perceptions of its environment.

$$\begin{array}{|l} \multicolumn{1}{l}{\textit{AutonomousAgentAct}} \\ \hline \textit{AutonomousAgent} \\ \textit{autoactions} : \mathbb{P}\,\textit{Motivation} \rightarrow \mathbb{P}\,\textit{Goal} \rightarrow \textit{View} \\ \qquad\qquad \rightarrow \textit{Environment} \rightarrow \mathbb{P}\,\textit{Action} \\ \hline \end{array}$$

Now, we need to define the state of an agent as follows by including the previous schemas and the current environment, and introducing variables for possible percepts, *posspercepts*, actual percepts, *actualpercepts*, and the actions performed, *willdo*.

$$
\begin{array}{|l}
\hline
\multicolumn{1}{l}{\mathit{AutonomousAgentState}} \\
\mathit{ObjectState} \\
\mathit{AutonomousAgentPerception} \\
\mathit{AutonomousAgentAct} \\
\mathit{env} : \mathit{Environment} \\
\mathit{posspercepts}, \mathit{actualpercepts} : \mathit{View} \\
\mathit{willdo} : \mathbb{P}\,\mathit{Action} \\
\hline
\mathit{actualpercepts} \subseteq \mathit{posspercepts} \\
\mathit{posspercepts} = \mathit{canperceive}\ \mathit{env}\ \mathit{perceivingactions} \\
\mathit{actualpercepts} = \mathit{willperceive}\ \mathit{motivations}\ \mathit{goals}\ \mathit{posspercepts} \\
\mathit{willdo} = \mathit{autoactions}\ \mathit{motivations}\ \mathit{goals}\ \mathit{actualpercepts}\ \mathit{env} \\
\hline
\end{array}
$$

1.3.4 Goal Generation

As stated above, in order for an agent to be autonomous, it must generate *goals* from *motivations*. The initial point in any interaction is when this goal generation process occurs. In this section, we describe how an autonomous agent, *defined* in terms of its somewhat abstract *motivations*, can construct goals or concrete states of affairs to be achieved in the environment. Our model requires a repository of known goals which capture knowledge of limited and well-defined aspects of the world. These goals describe particular *states* or *sub-states* of the world with each autonomous agent having its own such repository. An agent tries to find a way to mitigate motivations by selecting an action to achieve an existing goal or by retrieving a goal from a repository of known goals, as considered below.

In order to retrieve goals to mitigate motivations, an autonomous agent must have some way of assessing the effects of competing or alternative goals. Clearly, the goals that make the greatest positive contribution to the motivations of the agent should be selected. The *GenerateGoal* schema below describes at a high level an autonomous agent monitoring its motivations for goal generation. First, as indicated by *ΔAutonomousAgent*, the sociological agent changes, and a new variable representing the repository of available known goals, *goalbase*, is declared. Then, the motivational effect on an autonomous agent of satisfying a set of new goals is given. The *motiveffect* function returns a numeric value representing the motivational effect of satisfying a set of goals with a particular configuration of motivations and a set of existing goals. The predicate part specifies all goals currently being pursued must be known goals that already exist in the goalbase. Finally, there is a set of goals in the goalbase that has a greater motivational effect than any other set of goals, and the current goals of the agent are updated to include the new goals.

GenerateGoal

$$\begin{array}{l}
\Delta AutonomousAgent \\
goalbase : \mathbb{P}\, Goal \\
motiveffect : \mathbb{P}\, Motivation \rightarrow \mathbb{P}\, Goal \rightarrow \mathbb{P}\, Goal \rightarrow \mathbb{Z} \\
\hline
goals \subseteq goalbase \;\wedge\; goals' \subseteq goalbase \\
\exists\, gs : \mathbb{P}\, Goal \mid gs \subseteq goalbase \bullet \\
\quad (\forall\, os : \mathbb{P}\, Goal \mid os \in (\mathbb{P}\, goalbase) \bullet \\
\qquad (motiveffect\ motivations\ goals\ gs \geq \\
\qquad\qquad\qquad motiveffect\ motivations\ goals\ \ os) \wedge \\
\qquad goals' = goals \cup gs)
\end{array}$$

1.4 Inter-Agent Relationships

Agents and autonomous agents are thus defined in terms of goals. Agents *satisfy* goals, while autonomous agents may, additionally, generate them. Goals may be adopted by either autonomous agents, non-autonomous agents or objects without goals. Since non-autonomous agents satisfy goals for others they *rely* on other agents for purposeful existence, indicating that goal adoption creates critical inter-agent relationships.

A direct engagement takes place whenever a neutral-object or a server-agent adopts the goals of another. Thus, an agent with some goals, called the *client*, uses another agent, called the *server*, to assist them in the achievement of those goals.

DirectEngagement

$$\begin{array}{l}
client : Agent \\
server : ServerAgent \\
goal : Goal \\
\hline
client \neq server \\
goal \in (client.goals \cap server.goals)
\end{array}$$

Once autonomous agents have generated goals and engaged other server-agents, these server-agents may, in turn, engage other non-autonomous entities with the purpose of achieving or pursuing the original goal. This process can then, in principle, continue indefinitely. An *engagement chain* thus represents the goal and all the agents involved in the sequence of direct engagements. Since goals are grounded by motivations, the agent at the head of the chain must be autonomous.

$$
\begin{array}{|l}
\textit{EngagementChain} \\
\hline
goal : Goal \\
autonomousagent : AutonomousAgent \\
agentchain : \mathrm{seq}\, ServerAgent \\
\hline
\#agentchain > 0 \\
goal \in autonomousagent.goals \\
goal \in \bigcap \{ s : ServerAgent \mid s \in \mathrm{ran}\, agentchain \bullet s.goals \} \\
\hline
\end{array}
$$

A *cooperation* describes a goal, the autonomous agent that generated the goal, and those autonomous agents that have adopted that goal from the generating agent. In addition, all the agents involved have the goal of the cooperation, an agent cannot cooperate with itself, and the set of cooperating agents must be non-empty. Cooperation cannot, therefore, occur unwittingly between agents, but must arise as a result of the motivations of an agent and the agent recognising that goal in another. (The definition below does not capture the notion of this recognition and adoption but simply provides an abstract template which could be further elaborated as required.)

$$
\begin{array}{|l}
\textit{Cooperation} \\
\hline
goal : Goal; \; generatingagent : AutonomousAgent \\
cooperatingagents : \mathbb{P}\, AutonomousAgent \\
\hline
goal \in generatingagent.goals \\
\forall\, aa : cooperatingagents \bullet goal \in aa.goals \\
generatingagent \notin cooperatingagents \\
cooperatingagents \neq \{\, \} \\
\hline
\end{array}
$$

The key relationships in a multi-agent system are direct engagements, engagement chains and cooperations [19]. The combined total of direct engagements, engagement chains and cooperations (represented as *dengagements*, *engchains* and *cooperations*) defines a social organisation that is not artificially or externally imposed but arises as a natural and elegant consequence of our definitions of agents and autonomous agents. This organisation is defined in the *AgentSociety* schema below.

$$
\begin{array}{|l}
\textit{AgentSociety} \\
\hline
entities : \mathbb{P}\, Entity \\
objects : \mathbb{P}\, Object \\
agents : \mathbb{P}\, Agent \\
autonomousagents : \mathbb{P}\, AutonomousAgent \\
dengagements : \mathbb{P}\, DirectEngagement \\
engchains : \mathbb{P}\, EngagementChain \\
cooperations : \mathbb{P}\, Cooperation \\
\hline
\end{array}
$$

By considering the entire set of engagements, engagement chains and cooperations, a map of the relationships between individual agents can be constructed for a better understanding of their current social interdependence. The direct engagement relationship specifies the situation in which there is a direct engagement for which the first agent is the client and the second agent is the server. In general, however, any agent involved in an engagement chain engages all those agents that appear subsequently in the chain. To distinguish engagements involving an intermediate agent we introduce the indirect engagement relation *indengages*; an agent *indirectly* engages another if it engages it, but does not *directly* engage it. If many agents directly engage the same entity, then no single agent has complete control over it. It is important to understand *when* the behaviour of an engaged entity can be modified without any deleterious effect (such as when no other agent uses the entity for a *different* purpose). In this case we say that the agent *owns* the entity. An agent, *c*, owns another agent, *s*, if, for every sequence of server-agents in an engagement chain, *ec*, in which *s* appears, *c* precedes it, or *c* is the autonomous client-agent that initiated the chain. An agent *directly owns* another if it owns it and directly engages it. We can further distinguish the *uniquely owns* relation, which holds when an agent *directly* and *solely* owns another, and *specifically owns*, which holds when it owns it, and has only one goal. These definitions are presented below.

directly engages

$$\begin{array}{l} \forall\, c : Agent;\ s : ServerAgent \bullet (c, s) \in dengages \Leftrightarrow \\ \qquad \exists\, d : dengagements \bullet d.client = c \wedge d.server = s \end{array}$$

engages

$$\begin{array}{l} \forall\, c : Agent, s : ServerAgent \bullet (c, s) \in engages \Leftrightarrow \\ \quad \exists\, ec : engchains \bullet (s \in (\operatorname{ran} ec.agentchain) \wedge \\ \qquad\qquad\qquad c = ec.AutonomousAgent) \vee \\ \qquad\qquad (((c, s), ec.agentchain) \in before) \end{array}$$

indirectly engages

$$indengages = engages \setminus dengages$$

owns

$$\begin{array}{l} \forall\, c : Agent;\ s : ServerAgent \bullet (c, s) \in owns \Leftrightarrow \\ \quad (\forall\, ec : engchains \mid s \in \operatorname{ran} ec.agentchain \bullet \\ \qquad\qquad\qquad ec.AutonomousAgent = c\ \vee \\ \qquad\qquad ((c, s), ec.agentchain) \in before) \end{array}$$

directly owns

$$downs = owns \cap dengages$$

uniquely owns

$$\forall\, c : \mathit{Agent};\ s : \mathit{ServerAgent} \bullet (c, s) \in \mathit{uowns} \Leftrightarrow$$
$$(c, s) \in\ \mathit{downs} \wedge \neg\, (\exists\, a : \mathit{Agent} \mid a \neq c \bullet (a, s) \in \mathit{engages})$$

specifically owns

$$\forall\, c : \mathit{Agent};\ s : \mathit{ServerAgent} \bullet (c, s) \in \mathit{sowns} \Leftrightarrow$$
$$(c, s) \in\ \mathit{owns} \wedge \#(s.\mathit{goals}) = 1$$

Thus the agent framework allows an explicit and precise analysis of multi-agent systems with no more conceptual primitives than were introduced for the initial framework to describe individual agents. Using these fundamental forms of interaction, we can proceed to define a more detailed taxonomy of inter-agent relationships that allows a richer understanding of the social configuration of agents, suggesting different possibilities for interaction, as shown by the relationships below, taken from [20]. Importantly, the relationships identified are not imposed on multi-agent systems, but arise naturally from agents interacting, and therefore underlie all multi-agent systems.

1.5 Sociological Behaviour

Now, social behaviour involves an agent interacting with others; sociological behaviour requires more, that an agent understand its relationships with others. In order to do so, it must model them, their relationships, and their plans.

1.5.1 Models and Plans

To model their environment, agents require an *internal store*, without which their past experience could not influence direct behaviour, resulting in reflexive action alone. A store exists as part of an agent's state in an environment but it must also have existed *prior* to that state. We call this feature an *internal store* or *memory*, and define *store agents* as those with memories. Unlike *social* agents that engage in interaction with others, *sociological* agents model relationships as well as agents. It is a simple matter to define the model an agent has of another agent (*AgentModel*), by re-using the agent framework components as shown below. Even though the types of these constructs are equivalent to those presented earlier, it is useful to distinguish physical constructs from mental constructs such as models, as it provides a conceptual aid. We can similarly define models of other components and relationships so that specifying a sociological agent amounts to a refinement of the *Agent* schema as outlined below.

$$\mathit{EntityModel} == \mathit{Entity}$$
$$\mathit{AgentModel} == \mathit{Agent}$$
$$\mathit{AutonomousAgentModel} == \mathit{AutonomousAgent}$$
$$\mathit{CoopModel} == \mathit{Cooperation}$$
$$\mathit{EngModel} == \mathit{DirectEngagement}$$
$$\mathit{ChainModel} == \mathit{EngagementChain}$$

A sociological agent therefore views its environment as containing a collection of entities with engagements, engagement chains and cooperations between them. There are many consistency checks that need to be specified at this level, such as if a sociological agent has a model of a direct engagement (say) then it must model both those entities involved as having the goal of the direct engagement. However, there are many details, and we omit then in our definition below.

$$\begin{array}{lcl}
\mathit{ModelAgent} & ::= & \mathit{ent}\langle\!\langle \mathit{Entity}\rangle\!\rangle \\
 & \mid & \mathit{obj}\langle\!\langle \mathit{Object}\rangle\!\rangle \\
 & \mid & \mathit{agn}\langle\!\langle \mathit{Agent}\rangle\!\rangle \\
 & \mid & \mathit{aag}\langle\!\langle \mathit{AutonomousAgent}\rangle\!\rangle \\
\mathit{ModelRelationship} & ::= & \mathit{chain}\langle\!\langle \mathit{ChainModel}\rangle\!\rangle \\
 & \mid & \mathit{eng}\langle\!\langle \mathit{EngModel}\rangle\!\rangle \\
 & \mid & \mathit{coop}\langle\!\langle \mathit{CoopModel}\rangle\!\rangle \\
\mathit{Model} & ::= & \mathit{relmodel}\langle\!\langle \mathit{ModelRelationship}\rangle\!\rangle \\
 & \mid & \mathit{agentmodel}\langle\!\langle \mathit{ModelRelationship}\rangle\!\rangle
\end{array}$$

SociologicalAgent

$$\begin{array}{l}
\mathit{AutonomousAgentState} \\
\mathit{models} : \mathbb{P}\,\mathit{Model}
\end{array}$$

Now, in order to consider sociological agents with planning capabilities, we can construct a high-level model of *plan-agents* that applies equally well to reactive or deliberative, single-agent or multi-agent, planners. It represents a high-level of abstraction without committing to the nature of an agent, the plan representation, or of the agent's environment; we simply distinguish *categories* of plan and possible relationships between an agent's plans and goals. Specifically, we define *active* plans as those identified as candidate plans not yet selected for execution; and *executable* plans as those active plans that have been selected for execution.

Formally, we initially define the set of all agent plans to be a given set ([*Plan*]), so that at this stage we abstract out any information about the nature of plans themselves. Our highest-level description of a *plan-agent* can then be formalised in the *PlanAgent* schema below.

PlanAgent

$$\begin{array}{l}
\mathit{Agent} \\
\mathit{goallibrary} : \mathbb{P}\,\mathit{Goal} \\
\mathit{planlibrary}, \mathit{activeplans}, \mathit{executableplans} : \mathbb{P}\,\mathit{Plan} \\
\mathit{activeplangoal}, \mathit{plangoallibrary} : \mathit{Goal} \rightarrow\!\!\!\!\!\!+\; \mathbb{P}\,\mathit{Plan} \\
\hline
\operatorname{dom} \mathit{activeplangoal} \subseteq \mathit{goals} \;\wedge \\
\qquad \bigcup(\operatorname{ran} \mathit{activeplangoal}) = \mathit{activeplans} \\
\operatorname{dom} \mathit{plangoallibrary} \subseteq \mathit{goallibrary} \;\wedge \\
\qquad \bigcup(\operatorname{ran} \mathit{plangoallibrary}) \subseteq \mathit{planlibrary} \\
\mathit{goals} \subseteq \mathit{goallibrary} \;\wedge \\
\qquad \mathit{executableplans} \subseteq \mathit{activeplans} \subseteq \mathit{planlibrary}
\end{array}$$

```
── SociologicalPlanAgent ──────────────────────────
│ SociologicalAgent
│ PlanAgent
└──────────────────────────────────────────────────
```

The variables *goallibrary*, *planlibrary*, *activeplans* and *executableplans* represent the agent's repository of goals, repository of plans, active plans and executable plans, respectively. Each active plan is necessarily associated with one or more of the agent's current goals as specified by *activeplangoal*. For example, if the function contains the pair $(g, \{p_1, p_2, p_3\})$, it indicates that p_1, p_2 and p_3 are competing active plans for g. While active plans must be associated with at least one active goal, the converse is not true, since agents may have goals for which no plans have been considered. Analogously the *plangoallibrary* function relates the repository of goals, *goallibrary*, to the repository of plans, *planlibrary*. However, not necessarily all library plans and goals are related by this function.

1.5.2 Plan and Agent Categories

Now, using these notions, we can describe some example categories of goals, agents and plans (with respect to the models of the sociological plan-agent), that may be relevant to an agent's understanding of its environment. Each of the categories is formally defined below, where the sociological plan-agent is denoted as *spa*. Any variable preceded by *model* denotes the models that *spa* has of some specific type of entity or relationship. For example, *spa.modelneutralobjects* and *spa.modelowns* are the neutral objects and ownership relations the sociological agent models.

self-suff plan

$$\begin{aligned}&\forall p \in spa.planlibrary \bullet selfsuff(p) \Leftrightarrow spa.planentities(p) \subseteq \\ &\quad spa.modelneutralobjects \cup spa.modelself \cup \\ &\qquad spa.modelowns(\!|\, spa.modelself \,|\!)\end{aligned}$$

self-suff goal

$$\begin{aligned}&\forall g \in spa.goallibrary \bullet selfsuffgoal(g) \Leftrightarrow \\ &\qquad (\exists p \in spa.plangoallibrary(g) \bullet p \in selfsuff)\end{aligned}$$

reliant goal

$$\begin{aligned}&\forall g \in spa.goallibrary \bullet reliantgoal(g) \Leftrightarrow \\ &\qquad spa.plangoallibrary\ g \neq \{\ \} \ \wedge \\ &\qquad\qquad \neg\ (\exists p : spa.plangoallibrary\ g \bullet p \in selfsuff)\end{aligned}$$

A *self-sufficient plan* is any plan that involves only neutral-objects, server-agents the plan-agent owns, and the plan-agent itself. Self-sufficient plans can therefore be executed without regard to other agents, and exploit current agent relationships. (The formal definition uses the relational image operator: in general, the relational image

$R(\!| S |\!)$ of a set S through a relation R is the set of all objects y to which R relates to some member x of S.) A *self-sufficient goal* is any goal in the goal library that has an associated self-sufficient plan. These goals can then, according to the agent's model, be achieved independently of the existing social configuration. A *reliant-goal* is any goal that has a non-empty set of associated plans that is not self-sufficient.

For each plan that is not self-sufficient, a sociological plan-agent can establish the autonomous agents that may be affected by its execution, which is an important criterion in selecting a plan from competing active plans. An autonomous agent A may be affected by a plan in one of two ways: either it is required to perform an action directly, or it is engaging a server-agent S required by the plan. In this latter case, a sociological plan-agent can reason about either persuading A to share or release S, taking S without permission, or finding an alternative server-agent or plan. To facilitate such an analysis, we can define further categories of agents and plans, as described in [20], but we do not consider them further here.

1.6 Autonomous Interaction as Motivated Discovery

Many traditional models of interaction have assumed an ideal world in which unfounded assumptions have given rise to inadequate characterisations of interaction amongst autonomous agents. If we consider autonomous interaction to be a process of uncertain outcome (which it must be), then we can characterise it in a more general way as a process of *discovery* in terms of the effects of actions. This allows us to deal effectively with the inherent uncertainty in interaction. In this section, we show how the ideas of discovery can be used to approach autonomous interaction. We begin with an introduction to the ideas of discovery, and then show how they may be applied and formalised in the multi-agent domain.

1.6.1 Motivated Discovery

Scientific discovery is a process that occurs in the real world. Many examples of actual discovery have been observed and recorded, providing a basis for analyses of the reasoning methods used by real scientists. This has led to the identification of temporally and physically distinct elements in the discovery process which strongly support the notion of discovery as a methodology for reasoning rather than a single 'magical' process. Moreover, the underlying motivation behind scientific reasoning (and discovery) is one of increasing knowledge, understanding and awareness of a natural external environment in order to be able to explain, predict and possibly manipulate that environment. The second of these provides us with a large part of what we want to achieve in AI — to explain, predict and manipulate our environment. The first, if the notion of a methodology for discovery is even partly correct, provides us with a suitable means (in AI) for achieving it.

In the context of autonomous interaction, agents behave according to a certain understanding or *theory* of their environment and the agents within it, that we might in a different context consider to be the models we have just described. In this section, we will refer to a *theory* for the sake of adopting the *discovery* stance, but we might equally be referring to these models of others, and of interaction itself.

We adopt Luck's simple but encompassing model for discovery [16] that entails six stages, as follows.

1. **Prediction:** deductively generating predictions from a theory of interaction and a scenario;
2. **Experimentation:** testing the predictions (and hence the theory of interaction) by constructing appropriate experiments;
3. **Observation:** observing the results of experiments;
4. **Evaluation:** comparing and evaluating observations and predictions to determine if the theory has been refuted;
5. **Revision:** revising the theory to account for anomalies; and
6. **Selection:** choosing the best resulting revised theory.

The framework is a cyclical one, repeating until stability is achieved with a consistent theory. It begins with *prediction* which entails generating predictions for a given scenario, and then subjecting these to some kind of *experimentation*. Through *observation* and *evaluation*, the results of the experiment are compared with the predictions and, in the event that they are consistent with each other, no action is necessary. If the observations and predictions are anomalous, however, the theory must be *revised*, and a suitable revision *selected* to be passed through to the beginning of the cycle for use in generating new predictions. Even when no failure occurs, the theory is still liable to provide anomalies at a later stage.

Prediction

Perhaps the least troublesome part of the cycle is prediction. This is a simple deductive procedure that draws logical inferences from a theory and background knowledge given a description of a particular scenario. In order to make sense of our environment, we continually anticipate the effects of our actions, and of external factors — we make predictions about what will happen next. Usually, our predictions are correct and we anticipate well, but there are instances when the predictions fail, and we must deal with these failures later on in the cycle.

Generating predictions can be an expensive procedure, however, demanding time and resources which may not be available. We might for example be able to predict first, second and third places in an election, yet if we are only interested in who wins, only one of the predictions needs to be generated. This is related to the motivations of the reasoning agent, in the context of which the relevance of predictions can be assessed.

Experimentation

Once predictions have been generated, they may be empirically tested, and the results of these experiments can be compared with the predictions to determine if the theory (or indeed background knowledge) is, as much as possible, correct and consistent.

This implies a certain requirement on theories that has not yet been mentioned — that they be refutable, or *falsifiable*.

We can think of experimentation as being one of two kinds. First, there are *active* experiments in which the experimenter carefully constructs apparatus, or forces controlled environmental conditions with the aim of testing a particular characteristic or condition of a theory. Included in these are typical laboratory experiments. Alternatively, and more commonly, there are *passive* experiments which include any situation for which an expectation is generated, but for which there is no explicit theory. For example, squeezing a tube of toothpaste when brushing teeth is a passive experiment which has no controlled conditions, but which will determine if the expectation of producing toothpaste is correct or not. Both of these are important. When concerned with the problem of specifically acquiring knowledge in narrow domains, active experiments are prevalent. In normal everyday affairs, passive experiments are the norm unless they meet with a prediction failure. In this case, it is typical to switch to active experiments to find the reason for the failure, if necessary. In the case of autonomous interaction, any kind of act designed to elicit a response can be regarded as an experiment.

Thus experimentation is responsible for designing and constructing experiments in order that imperfections in the theory may be detected and corrected.

Observation

We intend this to be a complete and encompassing framework. Were we to exclude observation, it would not be so. Although observation immediately appears transparently simple, requiring merely that changes in the environment be observed and recorded for future reference, it is a little more complicated. (It should be noted that observations may be forced by the use of controlled experiments, or may occur independently.) Observations are compared with predictions and used to decide whether the theory is acceptable, or whether it needs to be revised.

Ideally, we would want an independent observer, a system capable of perceiving the external world, filtering out irrelevant information, and providing observations as input to the reasoning system.

Evaluation

At this point, the experiment has been carried out, the observations have been recorded, but it remains to decide whether or not the theory has been falsified, whether or not it is acceptable. To make this decision, we need to be aware of a number of influential factors and to evaluate the evidence in this light. Principally, this is concerned with the quality of the evidence. If an agent is to be effective, then it must be able to cope with both experimental and observational error, and must be able to evaluate them in an appropriate context. Little needs to be said about the occurrence of errors, for it is undeniable that they are always present to some degree. It is, however, unacceptable to pretend to cope with them by introducing simple tolerance levels. Experimental evidence must be evaluated relative to the current motivations of a system, taking into account the implications of success or failure. In medical domains, for example, even a small degree of error may be unacceptable if it would lead to the loss of a patient's life, while weather prediction systems may, in certain circumstances, allow a far greater error tolerance.

Revision

If it is decided that the theory has been falsified, then it must be revised so that it is consistent with the falsifying observations. Alternatively, new theories may be introduced or generated by another reasoning technique such as analogy, case-based reasoning, etc. The problem of creating new theories beyond direct observation is outside of this framework. Yet we do allow for their introduction into the inductive cycle, and in addition we allow for new theories based solely upon direct observation.

Revisions to the theory should include all those possible within the restrictions of the knowledge representation used that are consistent with the observations. This leads to the problem of combinatorial explosion, however, and the revision process should therefore be additionally constrained by heuristic search, the search heuristics being considered in the next and final stage. Allowing all revisions, potentially at least, is important in order that they are not pre-judged out of context.

Selection

As mentioned above, this is not really a separate stage, and proceeds in tandem with revision, but the task is distinct. Since the number of possible revisions to a given theory is extremely large, there must be criteria for selecting those theories which are better than others. Many criteria for rating theories have been proposed, such as simplicity, predictive power, modesty, conservatism and corroboration.

However, selection of theories must be in context. This means that the goals and motivations of a system are relevant to the task of judging which criteria are more important in evaluating a theory. The way in which these criteria are applied depends upon the context in which they are used and the need for which they are used. For appropriateness of use in many situations, we may prefer Newton's laws to Einstein's, but in other circumstances, only Einstein's may be acceptable.

1.6.2 Autonomous Interaction

In this subsection, we apply the framework just described to the problem of autonomous interaction, reformulating the discovery concepts, and formalising the process of interaction.

In order to make sense of our environment and to function effectively in it, we continually anticipate the effects of our actions and utterances — we make predictions (or expectations) about what will happen next. The action-selection function, *autoactions*, of the *AutonomousAgentAct* schema encompasses the deliberation of the agent. The action that is selected is intended to satisfy the goals of the agent through its resulting effects and consequent changes to the environment. In the case of an interaction episode involving two agents, the initiating agent selects an action that is intended to cause the desired response in the responding agent. The uncertainty inherent in such interaction means that the effects cannot be known in advance, but can only be discovered after the event has taken place, or action performed. We describe this by specifying the *predicted* effects of actions selected in the *AutonomousAgentAct* schema by applying the *socialeffectinteract* function to the current view of the environment and those actions. The agent thus predicts that these actions will change the environment to achieve the desired results. Remember that the environment includes all of the entities in it, so that a change to an agent in the environment will in turn cause a change to the environment itself. We also introduce a variable to store an agent's actual percepts prior to an operation, *oldpercepts*, which will be used later in the *Decide* schema.

Lastly, remember that *willdo*, specifies the actions that are performed and not the actions that the agent *selects* to perform next. In general, this is specified by the variable *todo* which is a function of the agents motivations, goals and current view of the environment (and *not* of the actual environment which dictates what actions are actually performed). In successful operation *willdo* should equal *todo*. However, when agents have only limited knowledge, perceptual capabilities or habiting dynamic open worlds these variables will not always equate. It is such anomalies that signal to the agent that it must revise its current model of itself and environment.

$$
\begin{array}{|l}
\textit{SocialAgentPredict} \\
\hline
\textit{SociologicalAgent} \\
\textit{socialeffectinteract} : \textit{View} \rightarrow \mathbb{P}\,\textit{Action} \nrightarrow \textit{View} \\
\textit{oldpercepts}, \textit{prediction} : \textit{View} \\
\textit{selectaction} : \mathbb{P}\,\textit{Motivation} \rightarrow \mathbb{P}\,\textit{Goal} \rightarrow \textit{View} \rightarrow \mathbb{P}\,\textit{Action} \\
\textit{todo} : \mathbb{P}\,\textit{Action} \\
\hline
\textit{todo} = \textit{selectaction motivations goals actualpercepts} \\
\textit{prediction} = \textit{socialeffectinteract actualpercepts todo} \\
\textit{prediction} \cap \bigcup \textit{goals} \neq \{\} \\
\hline
\end{array}
$$

In order to achieve the desired result, the relevant actions must be performed. Effectively, this acts as an experiment, testing whether the predictions generated are consistent with the resulting effects. In this sense, experimentation is central to this model, for such interaction with the environment is the only way in which an agent's understanding of its capabilities and its environment can be assessed to bring to light inadequacies, inconsistencies and errors. When an action is performed, it affects the models of other agents and of the agent society which, after the change, are derived from the previous models (of both the agents and inter-agent relationships) and the view of the environment through the function, *updatemodels*. These models of agents and their relationships are critical in determining if the action was successful.

$$
\begin{array}{|l}
\hline
\textit{SocialAgentInteract} \\
\textit{SocialAgentPredict} \\
\textit{updatemodels} : \mathbb{P}\,\textit{Model} \rightarrow \textit{View} \rightarrow \mathbb{P}\,\textit{Model} \\
\hline
\end{array}
$$

The action also has an effect on the environment, which changes accordingly, and a similar effect on the agent itself whose percepts also change. For example, in the case of an action which issues a request to another agent to tell the current time, the resulting model will either encode the fact that the agent is telling the time, or not. By inspecting this model and its attributes, the requesting agent can determine if its action has been successful. Note that the new value of *oldpercepts* takes the previous value of *actualpercepts* for later use.

$$
\begin{array}{|l}
\hline
\textit{SocialEnv} \\
\Delta \textit{SocialAgentPredict} \\
\textit{SocialAgentInteract} \\
\hline
\textit{env}' = \textit{effectinteract env willdo} \\
\textit{posspercepts}' = \textit{canperceive env}'\ \textit{perceivingactions} \\
\textit{actualpercepts}' = \textit{willperceive motivations goals posspercepts}' \\
\textit{willdo}' = \textit{autoactions motivations goals actualpercepts}'\ \textit{env}' \\
\textit{models}' = \textit{updatemodels models actualpercepts}' \\
\textit{oldpercepts}' = \textit{actualpercepts} \\
\textit{prediction}' = \textit{prediction} \\
\hline
\end{array}
$$

Evaluating the results of the actions appears simple. At the most basic level, it involves the comparison of predictions with observations. Thus if the intended effects of the actions and the actual effects match, then the actions have achieved the desired result and the episode is successful. If they are anomalous, then it reveals an erroneous understanding of the environment and the agents within it, or an inadequate capability for perception of the results. The important point here is that there is no guarantee of success, and failure can be due to any number of reasons.

This analysis assumes that the evidence is perfect, however, which may not always be appropriate. In any real environment this is not so, and error can be introduced into evidence in a variety of ways, reducing the quality of the observed evidence accordingly. Not only may there be inaccuracy due to the inherent uncertainty in both performing the actions and perception of the results (experimentation and observation respectively), but also, if the actions taken by the agent are communicative actions intended to elicit a response from another autonomous agent, then there may be inaccuracy due to malicious intent on the part of the responding agent by providing misleading information, for example [23]. Thus the response may itself be the vessel for the error.

In addition to assessing the fit of observations with predictions, therefore, the quality of the observations themselves must also be assessed in order to ascertain whether they are acceptable to be used in the comparison at all. Simple tolerance levels for assessing the acceptability of perceived evidence are inadequate, for they do not consider the need for the interaction episode, and the importance of achieving the desired result. The quality demanded of the observations can thus only be assessed in relation to the motivations of the agent which provide a measure of the importance of the situation, and take into account the implications of success and failure. In medical domains, for example, where the agents are highly motivated, even a small degree of error in interaction of relevant patient details may be unacceptable if it would lead to the loss of a patient's life, while neighbourly discussion of the weather with low motivations and little importance may allow a far greater error tolerance.

The schemas below describe evaluation with two predicates. The predicate *accept*, holds between the capabilities of the agent, its perceived environment before and after the actions were performed, and the agent models, if the evidence is acceptable. The capabilities of the agent capture the uncertainty information that arises from the agent itself, while the perceived environment and agent models include details of difficulties arising through the environment, or other agents. The *consider* predicate compares predictions and observations once evidence is accepted. Note that the potentially difficult question of when observations match predictions is bound up in the function itself which may be interpreted either as a simple equality test or as something more sophisticated.

The *Decide* schema also states at the beginning that though the agent changes as a result of this evaluation ($\Delta SocialAgent$), the state of the agent remains the same ($\Xi AutonomousAgentState$). Finally, if the evidence is accepted, and the observations do not match the predictions, then the agent models must be revised as specified by *revisemodels*.

$$bool ::= True \mid False$$

```
── SocialAgentEvaluate ──────────────────────
│ accept_ : (ℙ Action × View × View × ℙ Model)
│ consider_ : ℙ(View × View)
└────────────────────────────────────────────
```

```
── Decide ───────────────────────────────────
│ ΔSociologicalAgent
│ ΞAutonomousAgentState
│ SocialAgentPredict
│ SocialAgentEvaluate
│ revisemodels : View → ℙ Model → ℙ Model
├──────────────
│ accept (capabilities, actualpercepts, oldpercepts, models) ∧
│ ¬ consider (prediction, actualpercepts) ⇒
│     models′ = revisemodels actualpercepts models
└────────────────────────────────────────────
```

1.7 Discussion

Our efforts with BDI agents [10, 9] have shown that formal computational models of implemented systems and idealised systems, using the Z specification language [31], a standard (and commonly-used) formal method of software engineering, can result in implementations that are much more strongly related. In particular, they can be checked for type-correctness, they can be animated to provide prototype systems, and they can be formally and systematically refined to produce provably correct implementations. In this vein, related work has sought to contribute to the conceptual and theoretical foundations of agent-based systems through the use of such specification languages (used in traditional software engineering) that enable formal modelling yet provide a basis for implementation of practical systems.

Indeed, as the fields of intelligent agents and multi-agent systems move relentlessly forwards, it is becoming increasingly more important to maintain a coherent world view that both structures existing work and provides a base on which to keep pace with the latest advances. Our framework has allowed us to do just that. By elaborating the agent hierarchy in different ways, we have been able to detail both individual agent functionality and develop models of evolving social relationships between agents with, for example, our analyses of goal generation and adoption, and our treatment of engagement and cooperation. Not only does this provide a clear conceptual foundation, it also allows us to refine our level of description to particular systems and theories.

The problems with existing notions of agency and autonomy are now well-understood, but the importance of these notions remains high, nevertheless. In previous work we have addressed this by constructing a formal specification to identify and characterise those entities called agents and autonomous agents, in a precise yet accessible way. Our taxonomy provides clear definitions for objects, agents and autonomous agents that allow a better understanding of the functionality of different systems. It explicates those factors that are necessary for agency and autonomy, and is sufficiently abstract to cover the gamut of agents, both hardware and software, intelligent and unintelligent.

Then, by taking autonomous interaction to be a process of discovery within the framework, we can avoid the problems identified earlier of *guaranteed effects* and *automatic intention recognition*. In discovery, no effects are known for certain in advance, but instead, (tentative) predictions or expectations of future states of the world can be generated. It is only possible to be certain about effects once the actions have been carried out. This can lead to a re-evaluation of existing models.

Additionally, we assert that the process of autonomous communication must be motivated, and consequently a motivated agent does not have a *pre-determined agenda*, nor is it *benevolent*. Motivations provide a means by which an agent can set its own agenda, or set its own goals and determine which actions to perform in achieving them. The effects of benevolent behaviour are possible, but only through self-serving motivations. Moreover, because effects are not guaranteed, failure is always possible, but the combination of discovery and motivations allow effective exploitation of these failures and also recovery from them whenever possible.

Our aim in constructing the model for autonomous interaction is ambitious. We are attempting to provide a common unifying framework within which different levels of abstraction of reasoning, behavioural and interaction tasks can be related and considered. We have necessarily concentrated on a high-level specification so that the key principles can be explicated, but without sacrificing the need for preciseness through formality. By explicitly introducing motivated reasoning as part of the agent framework, and providing the capacity for effectively dealing with dynamic worlds through discovery, we provide a way in which the inadequacies in existing models may be addressed.

A significant claim of this work is that it can provide a general mathematical framework within which different models, and particular systems, can be defined and contrasted. Z is particularly suitable in squaring the demands of formal modelling with the need for implementation by allowing transition between specification and program. There are many well-developed strategies and tools to aid this transformation. Programs can also be *verified* with respect to a specification; it is possible to prove that a program behaves precisely as set out in the Z specification. This is not possible when specifications are written in modal logics since they have the computationally ungrounded possible-worlds model as their semantics. Thus our approach to formal specification is pragmatic; we need to be formal to be precise about the concepts we discuss, yet we want to remain directly connected to issues of implementation. We have also found the Z language is sufficiently expressive to allow a consistent, unified and structured account of a computer system and its associated operations.

References

1. Allen, J. *A plan-based approach to speech act recognition*. Technical Report 131, Department of Computer Science, University of Toronto, 1979.
2. Appelt, D. E. *Planning English Sentences*. Cambridge University Press. 1985.
3. Austin, J. L. *How to do Things with Words*. Oxford University Press. 1962.
4. Aylett, R., Brazier, F., Jennings, N., Luck, M., Preist, C. and Nwana, H. Agent systems and applications. *Knowledge Engineering Review*, **13**(3):303–308, 1998.
5. Boissier, O. and Demazeau., Y. A distributed artificial intelligence view on general purpose vision systems. In *Decentralized AI 3: Proceedings of the Third European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, E. Werner and Y. Demazeau, editors. Elsevier, 1992, pp 311–330.
6. Campbell, J. A. and d'Inverno, M. Knowledge interchange protocols. In *Decentralized AI: Proceedings of the First European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, Y. Demazeau and J. P. Müller, editors. Elsevier, 1990, pp 63–80.
7. Castelfranchi, C. Social power. In *Decentralized AI: Proceedings of the First European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, Y. Demazeau and J. P. Müller, editors. Elsevier, 1990, pp 49–62.
8. d'Inverno, M., Fisher, M., Lomuscio, A., Luck, M., de Rijke, M., Ryan, M. and Wooldridge, M. Formalisms for multi-agent systems. *Knowledge Engineering Review*, **12**(3):315–321, 1997.

9. d'Inverno, M., Kinny, D. and Luck, M. Interaction protocols in agents. In *ICMAS'98, Third International Conference on Multi-Agent Systems*, Paris, France, IEEE Computer Society, 1998a, pp 112–119.
10. d'Inverno, M., Kinny, D., Luck, M. and Wooldridge M. A formal specification of dMARS. In *Intelligent Agents IV: Proceedings of the Fourth International Workshop on Agent Theories, Architectures and Languages, (LNAI 1365),* M. P. Singh, A. Rao and M. J. Wooldridge, editors. Springer-Verlag, 1998b, pp 155–176.
11. Fisher, M. Representing and executing agent-based systems. In *Intelligent Agents: Theories, Architectures, and Languages, (LNAI 890),* M. Wooldridge and N. R. Jennings, editors. Springer-Verlag. 1995, pp 307–323.
12. Galliers. J. R. The positive role of conflicts in cooperative multi-agent systems. In *Decentralized AI: Proceedings of the First European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, Y. Demazeau and J. P. Müller, editors. Elsevier, 1990.
13. Galliers, J. R. Modelling autonomous belief revision in dialogue. In *Decentralized AI 2: Proceedings of the Third European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, Y. Demazeau and J. P. Müller, editors. Elsevier, 1991, pp 231–245.
14. Goodwin, R. A formal specification of agent properties. *Journal of Logic and Computation*, **5**(6):763–781, 1995.
15. Grice, H. P. Logic and conversation. In *Syntax and Semantics, Volume 3: Speech Acts*, P. Cole and J. L. Morgan, editors. Academic Press. 1975, pp 41–58.
16. Luck, M. Evaluating evidence for motivated discovery. In *Progress in Artificial Intelligence, (LNAI 727),* M. Filgueiras and L. Damas, editors. Springer-Verlag. 1993, pp 324–339.
17. Luck, M. From definition to deployment: What next for agent-based systems? *The Knowledge Engineering Review*, **14**(2):119–124, 1999.
18. Luck, M. and d'Inverno, M. A formal framework for agency and autonomy. In *Proceedings of the First International Conference on Multi-Agent Systems*, AAAI Press / MIT Press, 1995, pp 254–260.
19. Luck, M. and d'Inverno, M. Engagement and cooperation in motivated agent modelling. In *Proceedings of the First Australian DAI Workshop, (LNAI 1087),* Springer-Verlag, 1996, pp 70–84.
20. Luck, M. and d'Inverno, M. Plan analysis for autonomous sociological agents. In *Intelligent Agents VII, Proceedings of the Seventh International Workshop on Agent Theories, Architectures and Languages, (LNAI 1986),* Springer-Verlag, 2001a, pp 182–197.
21. Luck, M. and d'Inverno, M. A conceptual framework for agent definition and development. *The Computer Journal*, **44**(1):1–20, 2001b.
22. Luck, M., Griffiths, N. and d'Inverno, M. From agent theory to agent construction: A case study. In *Intelligent Agents III: Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, (LNAI 1193),* J. Müller, M. Wooldridge and N. R. Jennings, editors. Springer-Verlag, 1997, pp 49–63.
23. Marsh, S. Trust in distributed artificial intelligence. In *Artificial Social Systems: Selected Papers from the Fourth European Workshop on Modelling Autonomous Agents in a Multi-Agent World, (LNAI 830),* C. Castelfranchi and E. Werner, editors. Springer-Verlag. 1994, pp 94–114.
24. Norman, T. J. and Long, D. Goal creation in motivated agents. In *Intelligent Agents: Proceedings of the First International Workshop on Agent Theories, Architectures, and Languages, (LNAI 890),* M. Wooldridge and N. R. Jennings, editors. Springer-Verlag, 1995, pp 277–290.
25. Parsons, S., Sierra, C. and Jennings, N. Agents that reason and negotiate by arguing. *Journal of Logic and Computation*, **8**(3):261–292, 1998.

26. Perrault, C. R. An application of default logic to speech act theory. In *Intentions in Communication*, P. R. Cohen, J. Morgan and M. E. Pollack, editors. MIT Press. 1990, pp 161–186.
27. Rao, A. S. Agentspeak(l): BDI agents speak out in a logical computable language. In *Agents Breaking Away: Proceedings of the Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World, (LNAI 1038)*, W. Van de Velde and J. W. Perram, editors. Springer-Verlag, 1996, pp 42–55.
28. Rao, A. S. and Georgeff, M. P. An abstract architecture for rational agents. In *Proceedings of Knowledge Representation and Reasoning*, C. Rich, W. Swartout, and B. Nebel, editors. 1992, pp 439–449.
29. Searle, J. R. *Speech Acts*. Cambridge University Press. 1969.
30. Smith, R. G. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, **29**(12):1104–1113, 1980.
31. Spivey, J. M. *The Z Notation: A Reference Manual*. Prentice Hall, Hemel Hempstead, 2nd edition. 1992.
32. Wooldridge, M. J. and Jennings, N. R. Formalizing the cooperative problem solving process. In *Proceedings of the Thirteenth International Workshop on Distributed Artificial Intelligence*, 1994.