In ZUM'95: The Z Formal Specification Notation, J. Bowen and M. Hinchey, (eds.), Lecture Notes in Computer Science, 967, 47-62, Springer-Verlag, Heidelberg, 1995

Structuring a Z Specification to Provide a Formal Framework for Autonomous Agent Systems

Michael Luck[1] and Mark d'Inverno[2]

[1] Department of Computer Science, University of Warwick, Coventry, CV4 7AL, UK.
Email: mikeluck@dcs.warwick.ac.uk
[2] School of Computer Science, University of Westminster, London, W1M 8JS, UK.
Email: dinverm@westminster.ac.uk

Abstract. This paper describes a project which is using Z in the field of artificial intelligence (AI) to provide a defining framework for agency and autonomy. Specifically, the use of Z has provided a means for escaping from the terminological chaos surrounding agency and autonomy that is prevalent not just in the AI community, but also in other areas of computer science. We outline how we have developed a Z specification which serves as a framework that satisfies three distinct requirements. First, a framework should be defining in the sense that it must precisely and unambiguously provide meanings for the common concepts and terms. Second, it should be designed in such a way as to enable alternative models of particular classes of system to be explicitly described, compared and evaluated. Third, the framework should be sufficiently well-structured to provide a foundation for subsequent development of increasingly more refined concepts. The state based specification language Z is accessible to researchers from a variety of different backgrounds and allows us to provide a consistent unified formal account of an abstract agent system.

1 Introduction

1.1 Agency and Autonomy

The use of *agents* of many different kinds in a variety of fields of computer science and artificial intelligence is increasing rapidly. In artificial intelligence, the introduction of the notion of agents is partly due to the difficulties that have arisen when attempting to solve problems without regard to a real external environment or to the entity involved in that problem-solving process. Thus, though the solutions constructed to address these problems are significant, they are limited and inflexible in not coping well in real-world situations. In response, agents have been proposed as *situated* and *embodied* problem-solvers capable of functioning effectively in their environment.

It is now common for people to speak of *software agents*, *intelligent agents*, *interface agents*, *autonomous agents* and so on. The richness of the agent metaphor that leads to such different uses of the term is both a strength and a weakness. Its strength lies in the fact that it can be applied in very many different ways in many situations for different purposes. The weakness, however, is that the term *agent* is now used so frequently that there is no commonly accepted notion of what it is that constitutes an agent. For example, *agents* are often taken to be the same as *autonomous agents*, and the two terms are used interchangeably, without regard to their relevance or significance. The difference between these related, but distinct, notions is both important and useful in considering aspects of intelligence. Given the range of areas in which the notions and terms are applied, this lack of consensus over meaning is not surprising. As Shoham [7] points out, the number of diverse uses of the term *agent* are so many that it is almost meaningless without reference to a particular notion of agenthood.

It is now generally recognised that there is no agreement on what it is that makes something an agent, and it is standard, therefore, for many researchers to provide their own definition. In a recent collection of papers, for example, several different views emerge. Smith [8], takes an agent to be a "persistent software entity dedicated to a specific purpose." Selker [6] views agents as "computer programs that simulate a human relationship by doing something that another person could do for you." More loosely, Riecken [5] refers to "integrated reasoning processes" as agents. Others avoid the issue completely and leave the interpretation of their agents to the reader. In this paper, we report on a formal framework that we have constructed for *autonomy* and *agency* that attempts to bring together such disparate notions [4].

This section considers why we need a formal framework and what should be required of such formalisms. The second section introduces the key ideas contained in the framework and provides as overview of the specification which follows in the next section. Then we discuss the framework specification in relation to its usefulness in artificial intelligence and consider how it can be extended to incorporate architecture-specific ideas, illustrating this with the particular example of *knowledge interchange protocols*.

1.2 Formal Specification

In the current work, we have adopted the specification language Z [9] for two major reasons. First, it provides modularity and abstraction and is sufficiently expressive to allow a consistent, unified and structured account of a computer system and its associated operations. Such structured specifications enable the description of systems at different levels of abstraction, with system complexity being added at successively lower levels. Second, we view our enterprise as that of building programs. Z schemas are particularly suitable in squaring the demands of formal modelling with the need for implementation by allowing transition between specification and program. 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.

Furthermore, Z is gaining increasing acceptance as a tool within the artificial intelligence community (e.g. [3], [2]) and is therefore appropriate in terms of standards and dissemination capabilities.

Our concern has been to develop well-defined formal concepts that can be used both as the basis of an implementation, and also as a precise but general framework for further research.

1.3 Requirements of Formal Frameworks

We argue that a *formal framework* must satisfy three distinct requirements:

1. A formal framework must precisely and unambiguously provide meanings for common concepts and terms and do so in a readable and understandable manner. The availability of readable explicit notations allows a movement from vague and conflicting informal understandings of a class of models towards a common conceptual framework. A common conceptual framework will exist if there is a generally held understanding of the salient features and issues involved in the relevant class of models.

2. A framework should enable alternative designs of particular models and systems to be explicitly presented, compared and evaluated. It must provide a description of the common abstractions found within that class of models as well as a means of further refining these descriptions to detail particular models and systems.

3. The framework should be sufficiently well-structured to provide a foundation for subsequent development of new and increasingly more refined concepts. In other words, a practitioner should be able to choose the level of abstraction suitable for their purpose.

The use of abstraction renders prejudice about design unnecessary and enables a specification of a general system to be written. Z schema boxes are ideal for manipulation in the design process since by viewing the design process as a constraint of possible states, design strategies can be presented as further predicates in an abstract state schema.

2 A Framework for Agency and Autonomy: Overview

It was stated earlier that there is a lack of consensus over the meaning of the term agent and that there exist many diverse notions of agency. In this section, we introduce terms and concepts which are used to explicate our understanding of *agents* and *autonomous agents* and then developed into formal definitions.

Shoham [7] takes an *agent* to be any entity to which mental state can be ascribed. Such mental state consists of components such as beliefs, capabilities and commitments, but there is no unique correct selection of them. This is sensible, and we too do not demand that all agents necessarily have the same set of mental components. Indeed, we recognise the limitations associated with assuming an environment comprising homogeneous agents and consequently include in this discussion heterogeneous agents with varying capabilities. However, we do specify what is minimally *required* of an entity for it to be considered an agent in our framework.

We require a model that initially describes the environment and then, through increasingly detailed description, defines objects, agents and autonomous agents to provide an account of a general multiagent system. (This is what we mean by decreasing the level of abstraction in our descriptions of the world.) The definition of agency that follows is intended to subsume existing concepts as far as possible. In short, we propose a three-tiered hierarchy of entities comprising *objects*, *agents* and *autonomous agents*. The basic idea underlying this hierarchy is that all known entities are objects. Of this set of objects, some are agents, and of these agents, some are autonomous agents.

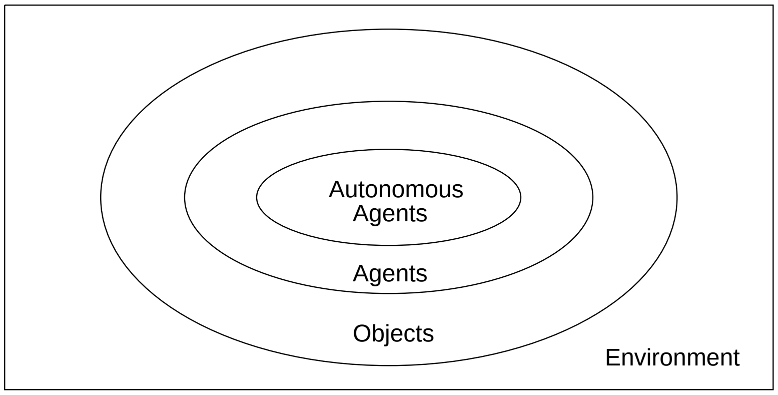

Fig. 1. The Entity Hierarchy

The specification had to be structured so that it reflected our view of the world as shown in the Venn diagram of Figure 1. It was required that the specification be built up in such a way that, starting from a basic description of an environment, each succeeding definition could be a refinement of the previously described entity. In this way, an object would be a refinement of the environment, an agent a refinement of an object, and an autonomous agent a refinement of an agent. Accordingly, the specification is thus structured into four parts.

Environment Our most abstract description of the world describes an environment simply as a collection of attributes.

Object Next, we cluster together some of the attributes in the environment and consider each such cluster as an object. Naturally, these objects are still collections of attributes, but we also give a more detailed description of these entities by describing their capabilities. The capabilities of an object are defined by a set of action primitives which can theoretically be performed by the object in some environment and, consequently, change the state of that environment.

Agent If we consider objects more closely, we can distinguish some objects which are serving some purpose or, equally, can be attributed some set of goals. This then becomes our definition of an agent, namely, an object with goals. With this increased level of detail in our description, we can define the greater functionality of agents over objects.

Autonomous Agent Further refinement of this description enables us to distinguish a subclass of the previously defined class of agents which are those agents that are autonomous. These autonomous agents are self-motivated agents in the sense that they follow their own agendas as opposed to functioning under the control of another agent. We thus define an autonomous agent as an agent with motivations and, in turn, show how these agents behave in a more sophisticated manner than their non-autonomous counterparts.

The Z specification language allowed just such a structured specification to be written by describing a system at its highest level of abstraction with further complexity being added at each successive lower level of abstraction. An overview of the structure of our framework is given in Figure 2, where the arrows represent schema inclusion. Definitions of *environment*, *object*, *agent* and *autonomous agent* are given by Env, Object, Agent and AutonomousAgent respectively. This shows how we constructed the most detailed entity description in the framework (of an autonomous agent situated in some environment) from the least detailed description (of an environment).

For objects, agents and autonomous agents, we define how they act in an environment in the schemas ObjectAction, AgentAction and AutonomousAgentAction respectively. For agents and autonomous agents, we detail how they perceive in a given environment in AgentPerception and AutonomousAgentPerception. For objects, agents and autonomous agents we define their state when situated in an environment in ObjectState, AgentState and AutonomousAgentState.

The framework is specified below. Whenever a new concept is introduced, a textual definition is given before its formal specification.

3 The Framework Specification

3.1 Environment

The first primitive that we need to define is an attribute. Attributes are simply features of the world and are the only characteristics which are manifest. They need not be perceived by any particular entity, but must be potentially perceivable in an omniscient sense.

Definition: An *attribute* is a perceivable feature.

$[\mathsf{ATTRIBUTE}]$

Definition: An *environment* is some collection of attributes.

```
┌─Env──────────────────────────────
│ Environment : ℙ ATTRIBUTE
└──────────────────────────────────
```

For the purposes of readability, we define a new type ENVIRONMENT as the power set of attributes.

$\mathsf{ENVIRONMENT} == \mathbb{P}\,\mathsf{ATTRIBUTE}$

3.2 Objects

At a basic level, an object can be defined in terms of its abilities and its attributes. An object, in this sense, is just a ‘thing’ or entity with no further defining characteristics. This provides us with the basic building block to develop our notion of agency.

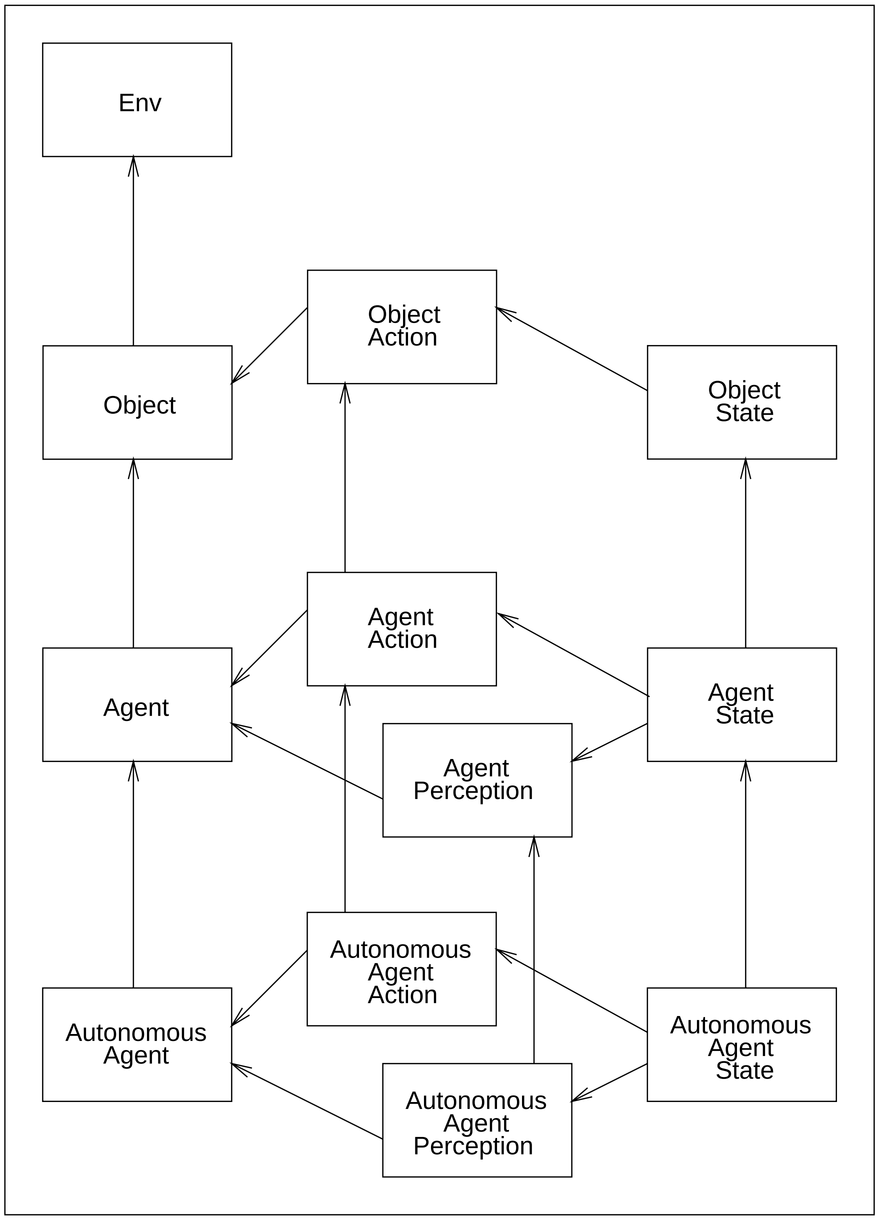

Fig. 2. Bottom-up View of the Use of Schema Inclusion in the Framework for Agency and Autonomy

Definition: An *action* is a discrete event which changes the state of the environment.

$[\mathsf{ACTION}]$

Definition: An *object* comprises a set of actions and a set of attributes.

An object in our model then is some subset of the environment to which we ascribe the notion of a set of basic capabilities.

$$\begin{array}{|l}
\hline
\mathsf{Object} \\
\mathsf{Env} \\
CapableOf : \mathbb{P}\,\mathsf{ACTION} \\
Attributes : \mathbb{P}\,\mathsf{ATTRIBUTE} \\
\hline
Attributes \subset Environment \\
\hline
\end{array}$$

Next, we define how an object will behave in an environment, determined by the function, $objectactions$, which states the set of actions which will be performed in a given environment. Notice that the predicate insists that any such actions will necessarily be within the capabilities of the object.

$$\begin{array}{|l}
\hline
\mathsf{ObjectAction} \\
\mathsf{Object} \\
ObjectActions : \mathsf{ENVIRONMENT} \rightarrow \mathbb{P}\,\mathsf{ACTION} \\
\hline
\forall\, env : \mathsf{ENVIRONMENT} \bullet (ObjectActions\ env) \subseteq CapableOf \\
\hline
\end{array}$$

Then for an object we define its state in a particular environment. We define a variable which represents the sequence of sets of actions that the object has previously performed, and a variable which gives the set of actions that the object will perform next.

In our model, we often wish to introduce terms which were totally dependent (mathematically) on other terms. We call such terms $redundant$ and any such terms presented in a schema are separated from the non-dependent state variables by use of a horizontal space.

$$\begin{array}{|l}
\hline
\mathsf{ObjectState} \\
\mathsf{Env} \\
\mathsf{ObjectAction} \\
History : \mathsf{seq}\ (\mathbb{P}\,\mathsf{ACTION}) \\
WillDo : \mathbb{P}\,\mathsf{ACTION} \\
\hline
\forall\, as : \mathsf{ran}\ History \bullet as \subseteq CapableOf \\
WillDo = ObjectActions\ Environment \\
\hline
\end{array}$$

Lastly, we show how an object actually behaves in an environment. When an object performs some set of actions, neither the basic capabilities, nor the action selection function will be changed.

ΔObjectState

ObjectState
ObjectState$'$

$$CapableOf' = CapableOf$$
$$ObjectActions' = ObjectActions$$

An *interaction* is simply that which happens when actions are performed in an environment. The effects of an interaction on the environment are determined by applying the $EffectInteraction$ function in the axiom definition below to the current environment and the actions taken. An axiom definition was chosen since in our model, all actions will result in the same change to an environment whether taken by an object, agent or autonomous agent.

$$EffectInteraction : \mathsf{ENVIRONMENT} \rightarrow \mathbb{P}\,\mathsf{ACTION} \nrightarrow \mathsf{ENVIRONMENT}$$

When an object interacts in its environment and performs an action, both the state of the object and the environment change. The history variable is updated by concatenating the current set of actions to the end of the sequence, and the new environment is given by applying $EffectInteraction$ to the old environment and the current actions. The last line states that the actions which follow the current actions are found by applying $ObjectActions$ to the new environment.

ObjectEnvInteract

ΔObjectState

$$History' = History \frown \langle WillDo \rangle$$
$$Environment' = EffectInteraction\ Environment\ WillDo$$
$$WillDo' = ObjectActions\ Environment'$$

3.3 Agents

We proceed to define agents in much the same way as for objects. There are many dictionary definitions for an agent. A recent paper by Wooldridge and Jennings [10] quotes *The Concise Oxford Dictionary* [11] definition of an agent as “one who, or that which, exerts power or produces an effect.” However, they omitted the second sense of agent which is given as “one who acts for another …”. This is important, for it is not the acting alone that defines agency, but the acting for *someone or something* that is the defining characteristic.

In this sense, agents are just objects with certain dispositions. They may always be agents, or they may revert to being objects in certain circumstances. An object is an agent if it serves a useful purpose either to a different agent, or to itself, in which case the agent is *autonomous*. This latter case is discussed further later. Specifically, an agent is something that ‘adopts’ or satisfies a goal or set of goals (often of another). Thus, if I want to store coffee in a cup, then the cup is my agent for storing coffee. It has been *ascribed* or has *adopted* my goal to have the coffee stored. It is, therefore, the goals of an agent which are its defining characteristics. We take a traditional AI view of goals as describable environmental states.

Definition: A *goal* is a state of affairs to be achieved in the environment.

$[\mathsf{GOAL}]$

Definition: An *agent* is an instantiation of an object together with an associated goal or set of goals.

The schema for an agent is simply that of an object but with the addition of goals.

```
┌─Agent──────────────────────────────
│ Object
│ Goals : ℙ GOAL
├──────────────
│ Goals ≠ { }
└────────────────────────────────────
```

Thus an agent has, or is *ascribed*, a set of goals which it retains over any instantiation (or lifetime). The same object may give rise to different instantiations of agents. An agent is instantiated from an object in response to another agent. Thus agency is *transient*, and an object may become an agent at some point in time, but may subsequently revert to being an object.

We now introduce perception. An agent in an environment may have a set of percepts available. These are the possible attributes that an agent could perceive, subject to its capabilities and current state. However, due to limited resources, an agent will not normally be able to perceive all those attributes possible, and bases action on a subset, which we call the *actual* percepts of an agent. Some agents will not be able to perceive at all. In the case of a cup, for example, the set of possible percepts will be empty and consequently the set of actual percepts will also be empty. The robot, however, may have several sensors which allow it to perceive. Thus it is not a requirement of an agent that it is able to perceive.

In our model we wish to distinguish between sets of attributes which represent a mental model of the world and those which represent the ‘actual’ environment. For clarity of exposition, we choose to define a type VIEW to be the perception of an Environment by an agent. This has an equivalent type to that of ENVIRONMENT, but now we can distinguish between physical and mental components of type $\mathbb{P}$ATTRIBUTE.

$\mathsf{VIEW} == \mathbb{P}\,\mathsf{ATTRIBUTE}$

It is also important to note that it is only meaningful in our model to consider perceptual abilities in the context of goals. Thus when considering objects which have no goals, perceptual abilities are not relevant. Objects respond directly to their environments and make no use of percepts even if they are available. We say that perceptual capabilities are *inert* in the context of objects.

In the schema for agent perception, AgentPerception, we add further detail to the definition of agency and so include the schema Agent. An agent has: a set of perceiving actions, $PerceivingActions$, which are a subset of the capabilities of an agent; a function, $CanPerceive$, which determines the attributes potentially available to an agent through its perception capabilities; and a function, $WillPerceive$, which describes those attributes which are actually perceived by an agent.

$$\begin{array}{|l}
\hline \textsf{AgentPerception} \\
\textsf{Agent} \\
PerceivingActions : \mathbb{P}\,\textsf{ACTION} \\
CanPerceive : \textsf{ENVIRONMENT} \rightarrow \mathbb{P}\,\textsf{ACTION} \nrightarrow \textsf{VIEW} \\
WillPerceive : \mathbb{P}\,\textsf{GOAL} \rightarrow \textsf{ENVIRONMENT} \rightarrow \textsf{VIEW} \\
\hline
PerceivingActions \subseteq CapableOf \\
\forall\, env : \textsf{ENVIRONMENT};\ as : \mathbb{P}\,\textsf{ACTION} \bullet \\
\qquad as \in \operatorname{dom}\,(CanPerceive\ env) \Rightarrow as = PerceivingActions \\
\operatorname{dom}\ WillPerceive = \{Goals\} \\
\hline
\end{array}$$

Directly corresponding to the goal or goals of an agent is an action-selection function, dependent on the goals, current environment and the actual perceptions. This is built up from the Agent and ObjectAction schemas.

$$\begin{array}{|l}
\hline \textsf{AgentAction} \\
\textsf{Agent} \\
\textsf{ObjectAction} \\
AgentActions : \mathbb{P}\,\textsf{GOAL} \rightarrow \textsf{VIEW} \rightarrow \textsf{ENVIRONMENT} \rightarrow \mathbb{P}\,\textsf{ACTION} \\
\hline
\forall\, gs : \mathbb{P}\,\textsf{GOAL};\ env1, env2 : \textsf{ENVIRONMENT} \bullet \\
\qquad (AgentActions\ gs\ env1\ env2) \subseteq CapableOf \\
\operatorname{dom}\ AgentActions = \{Goals\} \\
ObjectActions = AgentActions\ Goals\ Environment \\
\hline
\end{array}$$

The state of an agent includes two variables which describe those percepts possible in the current environment and a subset of these which are the current (actual) percepts of the agent in the environment.

$$\begin{array}{|l}
\hline \textsf{AgentState} \\
\textsf{AgentPerception} \\
\textsf{AgentAction} \\
\textsf{ObjectState} \\
PossiblePercepts : \textsf{VIEW} \\
ActualPercepts : \textsf{VIEW} \\
\hline
ActualPercepts \subseteq PossiblePercepts \\
PossiblePercepts = CanPerceive\ Environment\ PerceivingActions \\
ActualPercepts = WillPerceive\ Goals\ PossiblePercepts \\
(PerceivingActions = \{\ \}) \Rightarrow (PossiblePercepts = \{\ \}) \\
ObjectActions = AgentActions\ Goals\ ActualPercepts \\
\hline
\end{array}$$

We now define which of the $AgentState$ variables remain unchanged after a set of actions has been performed by that agent. If any of these variables ever did change, a different agent schema would have to be instantiated.

ΔAgentState

AgentState
AgentState′

$CapableOf' = CapableOf$
$Goals' = Goals$
$PerceivingActions' = PerceivingActions$
$CanPerceive' = CanPerceive$
$WillPerceive' = WillPerceive$
$AgentActions' = AgentActions$

The history and environment are altered in exactly the same way as previously described in the ObjectEnvInteract schema.

AgentEnvInteract

ΔAgentState
ObjectEnvInteract

3.4 Autonomous Agents

The definition of agency that we have developed above relies upon the existence of other agents which provide goals that are adopted in order to instantiate an agent. In order to ground the chain of goal adoption, to escape what could be an infinite regress, and also to bring out the notion of *autonomy*, we introduce *motivation*.

Definition: A *motivation* is any desire or preference that can lead to the generation and adoption of goals and which affects the outcome of the reasoning or behavioural task intended to satisfy those goals.

[MOTIVATION]

Definition: An *autonomous agent* is an instantiation of an agent together with an associated set of motivations.

AutonomousAgent

Agent
$Motivations : \mathbb{P}\,\mathrm{MOTIVATION}$

$Motivations \neq \{ \}$

The cup of the previous example cannot be considered autonomous because it cannot generate its own goals. By contrast, the robot is potentially autonomous in the sense that it may have a mechanism for internal goal generation, depending on its environment. Suppose the robot has motivations of achievement, hunger and self-preservation, where achievement is defined in terms of fixing tyres onto a car on a production line, hunger is defined in terms of maintaining power levels, and self-preservation is defined in terms of avoiding system breakdowns. In normal operation, the robot will generate goals to attach tyres to cars through a series of subgoals. If its power levels are low, however, it may replace the goal of attaching tyres with a newly-generated goal of recharging its batteries. A third possibility is that in satisfying its achievement motivation, it works for too long and is in danger of overheating. In this case, the robot can generate a goal of pausing for an appropriate period in order to avoid any damage to its components. Such a robot is autonomous because its goals are not imposed, but are generated in response to its environment.

The motivations, as well as the goals, of autonomous agents determine the way in which they perceive their environment. In the schema given below, the function, $AutoWillPerceive$, is then a more complex version of an agent's $WillPerceive$, but they are related — and must be since an autonomous agent is still an agent — as shown in the schema. However, that which an autonomous agent is *capable* of perceiving at any time is independent of its motivations. Indeed, it will always be independent of goals and motivations and there is consequently no equivalent increase in functionality to $CanPerceive$.

```
┌─ AutonomousAgentPerception ──────────────────────────
│ AutonomousAgent
│ AgentPerception
│ AutoWillPerceive : ℙ MOTIVATION → ℙ GOAL →
│                                   ENVIRONMENT → VIEW
├──────────────
│ WillPerceive = AutoWillPerceive Motivations
│ dom AutoWillPerceive = {Motivations}
└──────────────────────────────────────────────────────
```

We build up our schemas as follows:

```
┌─ AutonomousAgentAction ──────────────────────────────
│ AutonomousAgent
│ AgentAction
│ AutoActions : ℙ MOTIVATION → ℙ GOAL →
│                    VIEW → ENVIRONMENT → ℙ ACTION
├──────────────
│ dom AutoActions = {Motivations}
│ AgentActions = AutoActions Motivations
└──────────────────────────────────────────────────────
```

AutonomousAgentState

AutonomousAgentPerception
AutonomousAgentAction
AgentState

$WillPerceive = AutoWillPerceive\ Motivations$
$AgentActions = AutoActions\ Motivations$

Lastly, we specify the operation of an autonomous agent performing its next set of actions in its current environment. Notice that no explicit mention is made of any change in motivations, they may change in response to changes in the environment. If they do change, then the agent functions $WillPerceive$ and $AgentActions$ will also change. Further, motivations may generate new and different goals for the agent to pursue. In any of these cases, the characterizing features of an agent are in flux so that an autonomous agent can be regarded as a continually re-instantiated non-autonomous agent. In this sense, autonomous agents are permanent as opposed to transient non-autonomous agents.

ΔAutonomousAgentState

AutonomousAgentState
AutonomousAgentState$'$

$CapableOf' = CapableOf$
$PerceivingActions' = PerceivingActions$
$CanPerceive' = CanPerceive$
$AutoWillPerceive' = AutoWillPerceive$
$AutoActions' = AutoActions$

A description of an autonomous agent acting in the environment is built from that of an agent acting in the environment.

AutonomousAgentEnvInteract

ΔAutonomousAgentState
AgentEnvInteract

4 Evaluating the Framework

We have formal definitions for agents and autonomous agents which are clear, precise and unambiguous, but which do not specify a prescribed internal architecture for agency and autonomy. This is exactly right, since it allows a variety of different architectural and design views to be accommodated within a single unifying structure. All that is required by our specification is a minimal adherence to features of, and relationships between, the entities described therein.

Thus we allow a cup to be viewed as an object or an agent depending on the manner in which it functions or is used. Similarly, we allow a robot to be viewed as an object, an agent or an autonomous agent depending on the nature of its control structures. We do not specify here how those control structures should function, but instead how the control is directed.

The framework is suitable for reasoning both *about* entities in the world, and *with* entities in the world. That is to say that an agent itself can also use the entity hierarchy as a basis for reasoning about the functionality of other agents and the likelihood, for example, that they may or may not be predisposed to help in the completion of certain tasks.

4.1 A Foundation for Further Work

The framework described and specified here is intended to stand by itself, but also to provide a base for further development of agent architectures in an incremental fashion through refinement and schema inclusion. In this context, we are currently extending the framework to incorporate notions and mechanisms of communication through *knowledge interchange protocols*.

Knowledge interchange protocols (KIPs) provide scripts for communication between autonomous agents, first introduced by Campbell and d'Inverno [1]. While they have been used in many implemented systems, they seem to have been used in conflicting ways, possibly because a formal description has never been constructed. Essentially, KIPs provide a well-defined structure within which both agents and autonomous agents may communicate freely. If the communicating agents can agree on the reason for the communication, their dialogue follows a pre-compiled script or protocol, governed by the agreed purpose of the particular dialogue. Thus agents' contributions to a dialogue will depend on what is regarded as the purpose or intention of an exchange of information.

Each protocol can be seen as a sequence of general statements by agents taking part in the information exchange. For a particular instantiation of a protocol in a communication, a sequence of well-defined slots will be filled by the particular information that an agent has to contribute. Thus the actual amount of information communicated is minimised. There is no need for an agent to say things that are redundant because all agents know which protocol is being used, and the protocol itself ensures that risks of loops and long meaningless interactions are avoided. Communication overheads are thus minimised.

Here we give a very brief description of the specification of these protocols in relation to the framework described above.

In order to define protocols, we must show how actions can be structured and must therefore model the world at the $object$ level. First, we describe an object with the ability to communicate in that some subset of its capabilities are potentially *communicating actions*.

CommunicateObject

Object

$CommunicatingActions : \mathbb{P}\,\mathsf{ACTION}$

$CommunicatingActions \subseteq CapableOf$

The set of protocols is given by:

$[\mathsf{PROTOCOL}]$

A protocol object is defined as an object with communication capabilities, which has at its disposal a set of protocols which may be followed. Each protocol has associated with it a set of basic communicating actions.

$$\begin{array}{|l}
\hline \mathsf{ProtocolObject} \\
\mathsf{CommunicateObject} \\
Protocols : \mathbb{P}\,\mathsf{PROTOCOL} \\
ActionsOfProtocol : \mathsf{PROTOCOL} \rightarrow\!\!\!\!\!\!+\; \mathbb{P}\,\mathsf{ACTION} \\
\hline \mathrm{dom}\ ActionsOfProtocol = Protocols \\
\forall p : Protocols \bullet ActionsOfProtocol\ p \subseteq CommunicatingActions \\
\hline
\end{array}$$

An autonomous agent with a set of protocols is then as follows:

$$\begin{array}{|l}
\hline \mathsf{ProtocolAutonomousAgent} \\
\mathsf{AutonomousAgent} \\
\mathsf{ProtocolObject} \\
\hline
\end{array}$$

5 Conclusions

We have constructed a formal specification which identifies and characterises those entities that are called agents and autonomous agents. The work is not based on any existing classifications or notions because there is no consensus. Recent papers define agents in wildly different ways, if at all, and this makes it extremely difficult to be explicit about their nature and functionality. The taxonomy given here provides clear and precise 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, and so on.

Z has enabled us to produce a specification that is generally accessible to researchers in AI, as well as practitioners of formal methods. Through the use of schema inclusion, we are able to describe our framework at the highest level of abstraction and then, by incrementally increasing the detail in the specification, we add system complexity at appropriate levels. Our use of Z does not restrict us to any particular mathematical model, but instead provides a general mathematical framework within which different models, and even particular systems, can be defined and contrasted.

In particular, the nature of Z allows us to extend the framework and to refine it further to include a more varied and more inclusive set of concepts. The particular case of *knowledge interchange protocols*, by which the original framework was extended through new schemas and schema inclusion, indicates just how we intend to proceed in this respect, and how appropriate Z is for this task. It enables a practitioner to choose the level of detail required to present a particular design, and further provides an environment in which the design itself can be presented in increasing levels of detail.

Acknowledgements

Thanks to John Campbell, Jennifer Goodwin, Paul Howells, Colin Myers, Mark Priestley and John Wolstencroft for comments on earlier versions of this paper. The specification contained in this paper was checked for correctness using the *f*UZZ package.

References

1. J. A. Campbell and M. d'Inverno. Knowledge interchange protocols. In Y. Demazeau and J. P. Muller, editors, *Decentralized Artificial Intelligence*. Elsevier North Holland, 1989.
2. I. D. Craig. The formal specification of ELEKTRA. Research Report RR261, Department of Computer Science, University of Warwick, 1994.
3. R. Goodwin. Formalizing properties of agents. Technical Report CMU-CS-93-159, Carnegie-Mellon University, 1993.
4. M. Luck and M. d'Inverno. A formal framework for agency and autonomy. In *Proceedings of the First International Conference on Multi-Agent Systems*, 1995.
5. D. Riecken. An architecture of integrated agents. *Communications of the ACM*, 37(7):107–116, 1994.
6. T. Selker. A teaching agent that learns. *Communications of the ACM*, 37(7):92–99, 1994.
7. Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60:51–92, 1993.
8. D. C. Smith, A. Cypher, and J. Spohrer. Programming agents without a programming language. *Communications of the ACM*, 37(7):55–67, 1994.
9. J. M. Spivey. *The Z Notation*. Prentice Hall, Hemel Hempstead, 2nd edition, 1992.
10. M. J. Wooldridge and N. R. Jennings. Agent theories, architectures, and languages: A survey. In *Proceedings of the 1994 Workshop on Agent Theories, Architectures, and Languages*, 1994.
11. *The Concise Oxford Dictionary of Current English*. Oxford University Press, 7th edition, 1988.