*Journal of Experimental & Theoretical Artificial Intelligence*,
Vol. 15, No. 4, October–December 2003, 407–431

Taylor & Francis
Taylor & Francis Group

# Formal verification of multi-agent systems behaviour emerging from cognitive task analysis

ANA LILIA LAUREANO-CRUCES[1] and
AXEL ARTURO BARCELÓ-ASPEITIA[2]

[1] *Departamento de Sistemas, Universidad Autónoma Metropolitana, Azcapotzalco, Mexico*
e-mail: clc@correo.azc.uam.mx
[2] *Instituto de Investigaciones Filosóficas, Universidad Nacional Autónoma de México, Mexico*
e-mail: abarcelo@minerva.filosoficas.unam.mx

*Abstract*. The main goal of this paper is to present a novel formal approach to the verification of cognitive task analysis (CTA), an analytic tool that has been successfully used in the design of reactive behaviours, on multi-agent architectures. To achieve this, a formal logical system is developed, whose purpose is to formally check the possible success or failure of the resulting implementation. This logic's focus is on modelling an agent's behaviour based on her goals, perceptions and actions. The article starts by giving a brief introduction to current research in reactive systems and cognitive task analysis. Simple definitions are offered of the basic concepts in these fields: agent, object, reactive behaviour, control, etc. As illustration, the paper offers the results of applying CTA to a simple model of postal delivery. Then, the syntax and semantics of the proposed logic are defined. Finally, the logic is applied to the verification of some of the behaviours resulting of the previous CTA analysis.

*Keywords:* behaviour-based agents, cognitive task analysis, modal logic, formal language, artificial intelligence

*Received 10 June 2002; revised 19 February 2003; accepted 20 March 2003*

## 1. Introduction: agent based systems

These kinds of systems represent a new approach to distributed artificial intelligence that deals with problems of human behaviour simulation. This new way of behaviour modelling originated through 1980s research on mobile robots. As a result, its use has become widespread and it has been extended to solve different problems in the artificial intelligence area, such as intelligent teaching systems, expert systems, interface design, simulation and ecosystems. Hence, a good deal of theoretical research is being performed to model, design and implement agent based systems, also known as multi-agent (MA) systems.


ISSN 0952–813X print/ISSN 1362–3079 online 
http://www.tandf.co.uk/journals DOI: 10.1080/0952813031000119719

This paper deals with the analysis and design stages of MA systems. This is achieved using a formal language, showing that cognitive task analysis is an excellent tool for designing such systems. This paper assumes that the reader is familiar with agent based systems and basic knowledge of the subject can be found in Brooks (1991a), Maes (1993), Nwana (1996), Jennings (2000) and Laureano (1998, 2000). Nevertheless, a basic conceptual framework will be discussed, which will provide understanding of the methodology suggested in this work.

The crux of this analysis is to solve complex problems of the real world, leading to the development of robust and escalated software systems that require autonomous agents, which in turn are able to achieve their goals when they are situated in a dynamic uncertain environment.

### 1.1. *Differences and definitions between objects and agents*

It must be emphasized that a unique and pure definition of the 'agent' concept is not possible due to the very different braches of computer science (cognitive psychology, artificial intelligence and software engineering) that use it. Nevertheless, the general idea of the concept remains the same for all areas. The definition of the agent concept given by Jennings (2000) is as follows: 'An agent is an encapsulated computing system that is situated in an environment, which is able to show flexibility in the execution of autonomous actions that allows achievement of its goals'.

Besides, an agent must: (1) have the problem to solve perfectly defined, with its limits and interfaces well established (Laureano and de Arriga 1998, 2000); (2) be reactive and situated in a specific environment, having a good definition of the environment stimuli that will produce a reaction (Brooks 1991b, Laureano and de Arriaga 1999): (3) have specific goals that allow the system's global goal to be achieved (Nwana 1996); and (4) be autonomous, meaning that it is able to control its internal state and behaviour. Thus, agents have a flexible behaviour when solving problems in a dynamic environment.

The main characteristic of an agent is its autonomy, which provides the capacity of action within the environment in order to achieve its goals. In addition, this characteristic distinguishes them from objects. In other words, objects are able to encapsulate behaviour states, lacking the capacity to activate such behaviour. Accordingly, any object can call for the methods of other objects and, as a result, they are executed. In this sense, objects may be considered as software servers. Thus, they have no autonomy over their action capacity. On the other hand, agents know when they have to act or update their state.

Another characteristic of agents is that they can be seen as behaviour generators. For instance, a walking agent represents the walking behaviour. Similarly, the optimization agent is the one that optimizes a path. More example can be found in Brooks (1991a), Maes (1993), Laureano and de Arriaga (2000) and Laureano *et al.* (2001). Actually, the agent and object concepts are not new, since the theory emerged at the end of the 1970s. What is new is the proposal of analysis and design methodologies (Beer 1990, Atkinson-Abutridy and Carrasco 1999, d'Inverno and Luck 2000, Laureano *et al.* 2001) to deal with the implementation of MA systems, as far as they can be seen as behaviour generators. Hence, it is possible to say that agents are specializations of objects (Luck and d'Inverno 1995), where the object is the class and the specialization is the agent. The specialization consists of the level of autonomy that allows the agent to act in an environment based on received stimuli.

So far, the reaction characteristics of the agent to certain environment stimuli have been discussed, but it is necessary to clarify the environment concept. An environment is formed by objects and attributes. This means that an attribute is what an agent can sense according to its capabilities and situation. Therefore, attributes can be physical or abstract notions, according to the behaviour design. On the other hand, actions are developed to achieve agent goals, leading to appropriate modifications of the environment and therefore to proper alteration of the internal agent states (Laureano and de Arriaga 1998). In addition, MA systems can be formed by agents that, at the same time, can be organized into behavioural layers (García-Alegre *et al*. 1995).

MA systems' main features are: (1) the possibility to model complex systems in a distributed way, using a behaviour generation approach where different control locations are available; (2) the decentralization that offers less complexity in system's control, which in turn leads to a lower coupling between components; and (3) the control circuits of the action-selection, which are based on local statements of the problem to solve. Finally, it is necessary to emphasize that different types of agents can exist (Nwana 1996). However, there is a stronger tendency to avoid agent specialization and to design hybrid agents with different grades of: reactivity, cognition, and learning.

Therefore, a new design point of view can be formulated for distributed systems from what has been discussed so far, that is, the reactive behaviour design can be performed through the design and analysis of MA systems. A short definition of reactive behaviours along with their principal characteristics will be given in the next subsection. For a deeper study the reader is referred to Brooks (1991a, 1991b), Maes (1993), Laureano and de Arriaga (2000) and Beer (1990).

### 1.2. *Reactive behaviours*

The reactive behaviour trend is strongly influenced by behaviour psychology. Thus, implemented reactive agents can also be known as: (1) behaviour based agents, (2) situated agents or, simply, (3) reactive agents. Their behaviour involves decision making during execution using incomplete information based on simple rules of situated-action. Brooks is known as one of the main supporters of this trend, rejecting the necessity of an exhaustive symbolic representation (Brooks 1991b). Besides, he promotes the use of agent's reaction based on sensory inputs, which belong to partial views of the environment. Brooks states that the world is the best model to reason and this way he builds reactive systems based on perception and action (intelligence essentials).

Classic artificial intelligence decomposes an intelligent system in a functional way, that is, in a group of independent processors of information. The trend of reactive behaviours provides a guided decomposition oriented to behaviour activity. So, a group of activity processors (behaviours) are now available, which work in parallel and are connected to the real world through agent perception-action. Behavioural layers work individually and they can only extract those stimuli of the environment that are important for their performance. The environment is divided in sub-spaces or *situations*.

Behaviours are designed taking into account a sort of behavioural abstraction that allows sophisticated behaviours to be assembled of simple behaviours (one of the main ideas proposed by Brooks). The inferior layers are used to assemble basic

behaviours. For instance, basic robot behaviours can be to avoid hitting things, have wandering performance, etc. Superior layers are implemented to have abilities (goals) like delivering correspondence, looking for and gathering things, while wandering, and turning on lights.

### 1.3. *Control*

The control of this kind of system can be of several kinds; here are some examples:

- That of Brooks is based on two general mechanisms: inhibition and suppression. A control exists for each layer and superior layers include the control of the inferior layers when someone takes control. (architecture included). Each layer is able to substitute (to suppress) the inputs and to remove (to inhibit) the outputs of inferior layers; for pre-programmed finite intervals of time. The robot's ability (multi-agent architecture) of achieving its final goal, while the attention of the crucial goals continues in the inferior levels (monitoring process of critical situations), depends on the among-layers control programming, using the two previously mentioned mechanisms.
- In the architecture proposed by Kaelbling (1987), mediators are used. The mediators produce the desired combinations of the necessary agents; based on a certain stimulus.
- In the architecture of García-Alegre *et al.* (1995), the control is guided to tasks (Cañas and García-Alegre 1999). Each behavioural level counts with control cycles, goals and perceptions. When collisions of behaviours take place, the agent of the superior level will solve the problem. This control is guided to a specific task, that is, the agent of the superior level knows the situations that can be presented ahead of time and it knows who it should give the control according to its goal.

### 1.4. *Cognitive analysis of reactive behaviours*

Atkinson-Abutridy and Carrasco (1999) propose three different ways to design a reactive system: etiologic guide design, experimental design and situated action design.

Etiologic design bears in mind the fundamental steps of animal behaviour. The experimental design is an incremental bottom-up design to be tested and evaluated in parallel with the introduction of new behaviour patterns. The basic behaviour is the starting point of more sophisticated behaviours. The situated action design is based mostly on the agent's actions and its diverse behaviours according to the different situations of the environment. Consequently, identifying perceptions and actions is extremely important for reproducing situated behaviour.

Agent control, independently of the kind of design, will be based on two key aspects: behaviour representation and codification, and co-ordination among behaviours and environment. The case that interests us is to apply this method (CTA) to the design for located activity, of the basic behaviour of a postman. Laureano *et al.* (2001) gives a detailed description of its analysis and its resulting design. Cognitive task analysis was used to achieve this (Reeding 1992, Castañeda 1993, El Alami *et al.* 1998, Laureano and de Arriaga 1999), taking us to the result shown in figure 1.

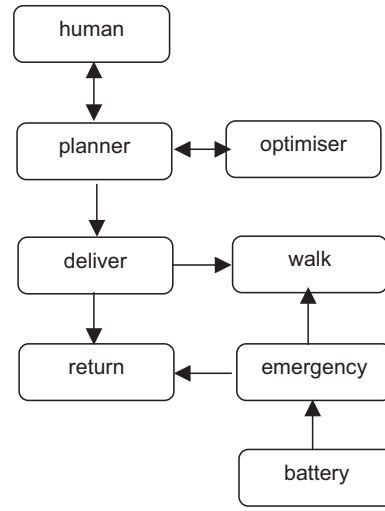


Figure 1. Activation and deactivation relationships among agents of the system. The arrows indicate which agents have the capacity to activate or to deactivate other agents.

## 2. Basic concepts

The following section offers basic definitions of the key concepts involved in CTA and its proposed formalization.

### 2.1. *Sources of information*

Information is related to the perception the agent has of itself and its environment. Perception is the mechanism through which the agent receives information. Sources of information form three different kinds:

- *Internal source*. This kind of information is comparable to humans' propioceptive perception (pains, sensations). This source of information is in the agent's body, which we may also call its immediate environment.
- *External source*. Reactive agents are located, which implies that external stimuli are in the environment and, according to the previous definitions, the best model would be to reason in the same world.
- *Internal memory*. Besides her body, the agent has privileged access to certain information stored in her memory.

In the postman's case, for example, cognitive behaviour development requires information exchange with sources of all three kinds: (1) from internal sources, through sensation, the postman system receives information on whether or not it is in physical condition to carry out the task (2) from external sources, it receives information regarding: (a) its physical location, (b) if there are any pending letters, (c) if they are still letters to deliver, (d) addresses for the houses and (e) the post office, and (f) addresses printed on the letters. All these external sources of information are resources the agent has in order to perform her task. Finally, (3) the internal memory serves (a) to know the location of the houses, (b) to elaborate and to follow a delivery plan; because the agent requires to keep information accessible in its memory.

2.2. *Universe*

The universe is constituted by objects, their attributes and the environment they are situated in.

(1) *Objects*: defined as those whose attributes can be affected by some kind of action. Their attributes are perceptible to other agents. In our example, objects are the letters and the postman. In the case of the letter, one of its attributes is the address, being this a source of external information. In the case of the postman, it is an object with goals, given the previous agent's conceptualization.
(2) *Environment*. Following intuitions from Barwise and Perry (1983), we conceive the environment as a frame of reference to locate objects in the world. Since every object is located in it, the environment will be modelled as a series of relations that assign information of different kinds to different objects. As example of this kind of information, think of the spatial location of the houses (which comes from the internal memory) or the fact that the agent or any other object, is in a certain spatial location (which comes from an external source of information). So, for example, the fact that the agent is at the post office will be modelled as establishing a relation between that object and its spatial location at a certain time. Intuitively, it could be modelled as a binary relations between two objects: the relation '$x$ is in $y$'.
(3) *Attributes*. An attribute is a piece of information. Of objects, only their attributes can be perceived, for example: the spatial location of an object, the addresses written in the letters and the delivery order inside the plan of the day, the space location of the post office and so with the rest of the relations that constitute the situation of an agent in its environment.
(4) *World*. A world is a complex system constituted by objects, their attributes and an environment determining their relations.
(5) *Facts*. Besides the components that constitute the environment, the world could be described by the facts that take place in it. Agents are constantly perceiving the environment and each perception models the state of the environment at a certain time, so constituting the facts. An object having an attribute constitutes a fact. Examples of facts in the postman's world can be: (a) an object being at certain place, (b) that the letter $x$ is directed to the address $y$, (c) that the agent postman is at the post office and (d) that the agent postman feels well. These are all different facts. Notice that the totality of relations within the environment determines the world, and vice versa.

2.3. *Agents*

In this section, we will introduce the terms and concepts necessary to explain and understand agents. Specification through languages as in the case of Z language (Spivey 1988) can be used to describe states and operations, considering the agent as an abstract machine that contains logic and an inference set.

Our starting point will be the *objects, agents* and *environment* concepts given by Luck and d'Inverno (1995). They utilize the Z language for grounding basic knowledge (Spivey 1988). According to Luck and d'Inverno, there is a hierarchy between object and agent where the agent is a specialization of the object. We will include a short summary. There is more information in specific references. This definition form has been used in the development of different multi-agent systems

(Jung 1998, Laureano and de Arriaga 1998). Next, some necessary basic definitions are presented for formalization. Thus, we may define an *Agent* as any object associated with one or more goals.

**Object**
Goals: P Goal
Goals $\neq \{\ \}$

An object could become an agent in serving the purposes of another agent or its own. Once its action ends, the agent becomes again an object.

It is necessary to keep in mind that *the goal of an agent* can exist in an explicit or implicit way. Hence, an agent's behaviour can be passive or active.

Passive: their goals are imposed or assigned.
Active: it refers to the case when an agent is able to change the state of the environment; developing actions that satisfy her goals, as in the case of a robot, an expert system or an ITS composed by agents.

### 2.4. *Agents in action*

So far, we have defined the main characteristics of an object and an agent. Now we will define the aspects of an agent in action. An agent's perception capability is fundamental because an agent acts upon what it sees and the quality of its action depending on the quality of perception able to be implemented. For an objective to be carried out by an agent, we must consider the following: its capabilities, what it is able to perceive based on these capabilities, and what it perceives at any given moment. We get the previous specification using three sub-functions: *PerceivingActions*, *CanPerceive and WillPerceive*. Next, we will define the functions that allow an agent to act. We will begin with the concept of view.

2.4.1. *View*. It is the perception that an agent has of the environment.

2.4.2. *Agent schema*. The results of CTA analysis can be given in what Luck and d'Inverno (1995) call *agent schema*. These schema condense the possible perceptions and actions that define an agent according to such analysis.

Perception OfAnAgent

| | |
|---|---|
| ActionsthatPerceive: | P Action<br>What an agent can do according to its capabilities. |
| WhatCanPerceive: | Environment $\rightarrow$ P Action $\mapsto$ Environment<br>These are the attributes that are potentially available according to the perceptual capabilities. |
| WhatWillPerceive: | P Goal $\rightarrow$ Environment $\rightarrow$ View<br>It represents the perceived attributes in a specific Moment, according to its goals. |

Additionally, we have the actions produced by the agent; these being the intentions of the agent that are represented by the actions that will be executed by a selection-action function depending on the goals, the state of the environment, and the perceptions at a specific moment.

**AnAgent'sAction**

| | |
|---|---|
| AnAgent'sActions: | P Goal → View → Environment → P Action<br>It guarantiees that the selection-action function obtains a set of actions belonging to the agent competence. |

2.4.3. *Actions*. In our model, an action is a group of changes inside the world. It is easy to see that because of the established framework, the action of a reactive agent may involve changes of two kinds:

(1) changes in the attributes of an object; or
(2) the elimination or addition of objects to the agent's situation.

Examples of (1) are: (a) location change of an object (because it involves the change of the information assigned by the location attribute to an object), (b) to assign information to an object (since it associates with the object information that it didn't have before). Examples of (2) changes are picking up, dropping objects, or when objects become active.

Actions that imply changes of both kinds exist. To move in space, for example, involves a change of kind (1); since it affects the value assigned by the location relation. The changes of kind (2) are presented when changing location; since the objects that conform their situations change. In normal situations, the incorporation of an object to the world must come accompanied by a series of changes of kind (1) that assign to the new object certain information that makes it excellent to the task in question. These three elements—sources of information, environment and actions—are enough to model the development of a cognitive task.

We give to both kinds of changes a homogenous logical treatment based on the evolution of objects modelled in Bertino *et al.* (1999) for object databases. This will be developed in more detail in the semantics of section 4.2.

2.4.4. *Behaviour*. The use of the term 'behaviour' in this article coincides with the standard use in theoretical artificial intelligence, at least, since Turing started using this term in 1936. Even though there is no definitive explicit definition of behaviour in the artificial intelligence literature, there is a recognizable tacit agreement about what aspects of behaviour are important for successful computer modelling. Gregory, for example, defines behaviour as any problem that can be solved by instructions, rules or explicit procedures (1987: 784), while Churchland and Churchland (1990: 26) define it as any systematic pattern of reaction to the environment in terms of a rule-governed input-output function. In both cases, behaviour is defined in terms of rules, procedures or explicit functions.

2.4.5. *Interaction*. An *interaction* is a set of actions on a common environment. The effect of an interaction on the environment is determined as a function of its initial state and subsequent actions.

| | |
|---|---|
| *EffectOfInteraction:* | *Environment* → *P Action* ↦ *Environment* |
| where: | → represents the correspondence of one domain to another.<br>↦ represents the partial correspondence of a domain to another. |

## 3. Behaviour agent schemes

The schemes of the agents, which conform the postman behaviour, are now defined using the information given in previous sections.

**Planner agent**

| | |
|---|---|
| *Goal:* | make a delivery path considering input attributes |
| *Class of Goal:* | active. |

*Perception*

| | |
|---|---|
| *ActionsthatPerceive:* | follow a path to come back home. |
| *WhatCanPerceive:* | walking. |
| *WhatWillPerceive:* | walk back from any point where delivery has been accomplished. |

*Action*

| | |
|---|---|
| *AnAgent'sActions:* | supply to the delivery agent the path that must be followed. |

**Optimiser agent**

| | |
|---|---|
| *Goal:* | select an optimum path based on the input data given by the planner agent. |
| *ClassOfGoal:* | passive (imposed by planner). |

*Perception*

| | |
|---|---|
| *ActionsthatPerceive:* | make an optimum delivery way based on: addresses and priority order. |
| *WhatCanPerceive:* | offices. |
| *WhatWillPerceive:* | offices with mail at a given time. |

*Action*

| | |
|---|---|
| *AnAgent'sActions:* | give to the delivery agent the optimum way to deliver. |

**Delivery agent**

| | |
|---|---|
| *Goal:* | deliver mail. |
| *ClassOfGoal:* | Passive (imposed by planner). |

*Perception*

| | |
|---|---|
| *ActionsThatPerceive:* | deliver it. |
| *WhatCanPerceive:* | walk and the offices. |
| *WhatWillPerceive:* | walk and arrive to the points selected by the optimiser agent. |

*Action*

| | |
|---|---|
| *AnAgent'sActions:* | walk and deliver mail. |

**Return agent**

| | |
|---|---|
| *Goal:* | Return to the beginning point. |
| *ClassOfGoal:* | active (once the delivery ends, the return behaviour is activated) or passive (if the emergency behaviour is activated). |

*Perception*

| | |
|---|---|
| *ActionsThatPerceive:* | follow the optimum path to beginning point. |
| *WhatCanPerceive:* | walking. |

*WhatWillPerceive:* walk back from any point where delivery has been accomplished.

*Action*

*AnAgent'sActions:* walk back to the beginning point.

**Emergency agent**

*Goal:* recharge battery and finish delivery.

*ClassOfGoal:* active (it is activated when the energy of batteries runs out)

*Perception*

*ActionsThatPerceive:* (a) go to recharge point, and (b) return and recharge battery before finishing delivery.

*WhatCanPerceive:* (a) analyse the energy input data supplied by sensors and (b) know whether it can continue or it needs a recharge.

*WhatWillPerceive:* reading of input data supplied by sensors at any given time.

*Action*

*AnAgent'sActions:* (a) path generation from the postman position to the point of recharge, (b) recharge and (c) conclude delivery.

**Walk agent**

*Goal:* move from one point to another without accidents (crash, fall or run out of batteries).

*ClassOfGoal:* Passive (imposed by delivery agent, return agent or emergency agent).

*Perception*

*ActionsThatPerceive:* wandering.

*WhatCanPerceive:* wander in the environment-universe.

*WhatWillPerceive:* wander in the next environment (that involves the delivery path).

*Action*

*AnAgent'sActions:* movements according to the goals of the: return agent or delivery agent or emergency agent.

**Battery agent**

*Goal:* knowledge about energy levels.

*ClassOfGoal:* active (each cycle of time is activated when motion begins).

*Perception*

*ActionsThatPerceive:* reading of input data supplied by sensors.

*WhatCanPerceive:* analysis of vital information.

*WhatWillPerceive:* reading of input data supplied by sensors at a given time interval.

*Action*

*AnAgent'sActions:* understand a set of input data. If operation trespass security limits, the emergency agent is informed about this situation.

Based on the previous definitions of behaviours for each one of the agents, which integrate the ability to deliver mail, a formal semantic language will be developed that will allow to prove that an appropriate correspondence of information exists among agents; warranting the success of the resulting capacity (postman delivery).

## 4. Logic

The development of formal logic in the last century has endowed artificial intelligence with one of its most potent and rigorous tools. Originally developed for the analysis of mathematical proofs, formal logical systems have become standard instruments for the analysis of reasoning in general.

Given that one of the central goals of artificial intelligence is the modelling and mechanization of human reasoning, the importance of the results and tools of formal logical for our discipline is obvious (Gabbay 1993). Indeed, in the last few decades, the interaction between logic and artificial intelligence has multiplied as much in its extension as in its importance. At the moment, logic is used in artificial intelligence as a tool for, among other things, theoretical foundations (Turing 1947, 1950), systematic representation of information and knowledge (Newell 1981, Mylopoulos *et al.* 1984), and even as imperative programming language. This interaction has affected not only artificial intelligence, but also logic. New developments in logic, like situation theory (Barwise and Perry 1983), have arisen with the explicit purpose of assisting the practice of artificial intelligence. Besides, this application, has given a new relevance to areas like epistemic logic (Meyer and van der Hoek 1995), modal logic (Fischer and Ladner 1977, Van Benthem 1989), dynamic logic (Fischer and Ladner 1979, Parikh 1981) and linear logic (Girard 1987).

Logic is a very valuable tool whose analytic power allows us to deepen in the results of cognitive task analysis with the maximum rigor and ease. For this purpose, we have developed a logical and mathematical model of the behaviour of multi-agent systems, in the tradition of computational semantics.

### 4.1. *Syntax*

4.1.1. *Vocabulary*. Besides the traditional vocabulary of first order modal logic with identity—(1) the traditional logical constants $\Box$, $\Diamond$, $\forall$, $\exists$, $\neg$, $\wedge$, $\vee$, $\supset$, $=$, (,), (2) object variables of the form $x_i$, (3) predicative constants of the form $P_i^n$ and (4) a series of constant of the form $c_i$—the vocabulary of our language contains a series of information variables of the form $y_i$ and a couple of new logical constants: the conditional deontic arrow and the arrow $\Rightarrow$ of objects-exchange $\rightarrow$.

4.1.2. *Formulae*. In this section, we will define the different expressions that form an instruction. First, any well formed formula $A$ (open or closed) in the first order fragment of our language is called a *factual formula*. These formulae have the capacity to express facts, as defined in sections 3.3.5 and 6.1. Besides, we include as *formulae of objects-exchange* those expressions of the form $\{t_{i1}, t_{i2}, \ldots t_{in}\} \rightarrow t_{i(n+1)}, t_{i(n+2)}, , \ldots t_{i(n+k)}\}$ such that $t_{i1} \neq t_{i2}, t_{i2} \neq t_{i3}, \ldots$ and $t_{i(n+k-1)} \neq t_{i(n+k)}$. These formulae receive their name, because the $t_n$ terms that occur in them refer to objects that are exchanged between the agent and its environment as a result of a particular action. Given a factual formula $A$ and a formula of objects-exchange $B$, we call *action formulae* any expressions of any of the following forms: $A$, $(A, B)$ or

*B*. Whithin a formula of action of the form $(A, B)$, we call $A$ its *factual component*, and $B$ its content in terms of object change.

The role of factual formulae in the language is to express the necessary conditions for an action according to an instruction (see definition of instruction in 4.1.3) and, in case of being the factual component of an action, the fact that results from the same action. In the same way, an objects-exchange formula plays the role of determining which objects of the world come into the agent's direct active power and become part of her informational situation as a result of an action. (In section 4.2.1, we will explain how this set incorporates the basic intuition behind Situation Theory without requiring a commitment to a full fledged situation semantics.)

4.1.3. *Instruction*. Finally we are in position to define what an instruction is. An instruction is a conditional formula of the form $A \Rightarrow B$ where $A$ is a factual formula and $B$ a formula of action. The intuition on which this definition rests is that every instruction is a conditional imperative. In this sense, an instruction $\mathrm{A} \Rightarrow \mathrm{B}$ tells the agent to carry out action B, if the condition A is satisfied—given the available information and in accordance to the agent's cognitive capacities. Consider the following couple of examples: (a) 'if there are letters to be delivered and energy in battery, make a delivery plan' would be symbolized as '$(\exists x \ \ letter(x) \wedge read(battery \ \ sensor) = charged) \Rightarrow \exists y \ \ delivery \ \ plan(y)$', (b) 'continue delivery while there is still letters to deliver and energy in battery' is symbolized as '$(\exists x \ \ letter(x) \wedge read(battery \ \ sensor) = charged) \Rightarrow status(delivery) = active$', etc.

4.1.4. *Program for MA systems*. Let a MA systems $\boldsymbol{S}$ be a non-empty set of agents $\{\alpha_1, \alpha_2, \alpha_3, \ldots \alpha_\mathrm{n}\}$. Define the *program* of the system as the structure. $\boldsymbol{P_S} = \langle\langle \mathrm{P}_1, \mathrm{P}_2, \mathrm{P}_3, \ldots \mathrm{P}_\mathrm{n}\rangle, A\rangle$ where each $\mathbf{P_i}$ is a set of instructions corresponding to each agent $\alpha_\mathrm{i}$ of system S, and $A$ is a factual modal formula expressing the program's goal. The function of formula $A$ is to describe the state of the universe expected after the program has successfully finished. Notice that the function of formula $A$ is not simply to describe the desired state of the world at the end of the program's execution. In many cases, it may be necessary to include within the criteria of a cognitive task's success, not only the final state of the world, but also particular actions of the agent or intermediate states of the world. For example, in the postman's task that we have been using to illustrate our analysis, the goal is to deliver the letters. The success at the end of this task is proved by checking two kinds of conditions. One is to examine the state of the world when the task is finished in order to check that, for example, every letter has reached its actual destination. However, there are other conditions where satisfaction cannot be verified in this simple way. For example, the task may be considered successfully accomplished even if the mail is only partially delivered; either because the addresses could not be reached or for some other justified reason. In this case, it is not enough to examine the state of the world at the end of the program. It is also necessary to examine the intermediate states through which the system went. In order to consider conditions of this sort, it is necessary to extend our logic beyond first order an incorporate a modal element. In this sense, our logic becomes a kind of temporal logic (Galton 1995, Van Bentham 1983).

4.2. *Semantics*

The goal of the following formal semantics is to provide a clear and mathematically decidable approach to verifying the success or failure of a behavioural cognitive analysis. The semantic proposal is an algebraic frame of possible worlds. The use of this kind of semantics for a formal system like ours is suggested by the combination of modal (for checking the task's final success), dynamic (in the instructions) and first order elements (at the moment of expressing action conditions within the instructions). Possible world semantics has proved to be flexible enough to provide semantic structures for such heterogeneous logics (Hustadt *et al.* 2001). This kind of semantics has been used to model several agent properties (Baader and Ohlbach 1995, Nwana *et al.* 1996, Wolter and Zakharyaschev 1998) as well as multi-agent systems (Manna and Pnueli 1992, Ndumu and Nwana 1999, Alvarado and Sheremetov 2001, Hustadt *et al.* 2001). However, there is still a lot to do in this field. Particularly, this kind of analysis has not yet been extended to the field of cognitive task analysis.

Given that our logic combines elements of dynamic and temporary logic, it is necessary to take position in the current debate regarding the relation between time and action. In this sense, we subordinate time to action, in such a way that it is actions that generate time, instead of taking place in it (Pratt 1976, Harel 1984, Galton 1987, Van Bentham 1983).

4.2.1. *What have we learnt from situation theory?* As said before, our logic incorporates certain intuitions from Situation Theory. However, its semantics is a possible world semantics. Hence, it is important to clarify its relation to situation theory.

Perry and Barwise introduced Situation Theory in the early eighties as an alternative to possible world semantics for natural language (Barwise 1989, Barwise and Perry 1981, 1983), specially regarding epistemic problems (Barwise 1988). It was later developed into a full fledged theory of information (Barwise and Seligman 1997). In explaining his original motivation, Barwise complained about the 'dubious metaphysical and epistemological assumptions' (1989: 7) upon which possible world semantics was built. In particular, Barwise was weary of the problem of 'omniscience' and the ontological status of possible worlds.

The basic intuition behind situation theory is that no agent is omniscient. In other words, no agent is in direct informational relation with the universe as a whole. Cognitive agents like us not only never know everything there is to know about the world, but also are in no position to find out anything they want about it.[1] In any given situation, an agent has only certain information at her disposal. The information accessible to an agent depends on the situation she is in. Thus, what is relevant for modelling an agent's informational state is not the actual state of the whole world, but only of her situation. Thus, situations must be seen as partial descriptions of the world.

So far, we agree wholeheartedly with Barwise and Perry's intuition and recognize its importance. However, we also believe that, when trying to model the behaviour of reactive agents in a system, this simple intuition can be easily incorporated into the formal framework of possible world semantics. In no way must this be taken as a criticism of the sophisticated and subtle theory that Barwise and others have developed out of this intuition. It has proved to be a very helpful tool in the analysis of very complex informational phenomena in linguistics and computer science (Cooper *et al.* 1990). Furthermore, its philosophical significance for our under-

standing of knowledge, meaning and information is also beyond doubt. Nevertheless, such complexities are aimed at better explaining information flow involving representation (Barwise and Seligman 1997). Reactive agents, on the contrary, do not exchange information through inner representations. This is one of their most important characteristics. Hence, most of the technical apparatus of situation semantics is irrelevant to their study.

Thus, the challenge is to incorporate situations into a possible world semantics. We accomplish this by requiring that the information necessary for the application of an instruction in an agent's program must be information accessible to such agent. In other words, it is not sufficient that such information be true in the world. It must also be available in the agent's situation. An instruction that, for example, asks an agent $\alpha$ to perform certain action $a$ conditional to certain fact $f$, should not be merely interpreted as a conditional statement saying that if $f$ is true, $\alpha$ must perform $a$. Instead, it says that only if the information required to determine the truth of $f$ is available to $\alpha$ (in her situation), the aforementioned conditional is the case. This is the reason why it is so important for Cognitive Task Analysis to determine the perceptual constrains of an agent.

Thus, we incorporate the intuition behind Barwise's 'situations' into our formal semantics through agent specific *domains*. An agent's domain includes only those objects of the world that are in the agent's possession. These objects are the only ones from which the agent can obtain information. The information available to an agent is fully determined by the attributes of the objects of her domain (section 2.4.3). It may be argued that our domains are nothing but situations under a different name, so that by incorporating them in our semantics, we are actually using a situation semantics. That would be fine for our purposes, because we do want to incorporate the advantages of situation semantics into our proposal, indeed.

On the other hand, there are also multiple advantages for keeping possible worlds in our model. For example, since it is vital for our goals to model the effects of every agent's behaviour (because it is through them that we judge the successful performance of the task, and also because reactive agents respond to changes in their environment created by other agents in the system), it is preferable to be able to keep full fledged possible worlds in our semantics. Many times, an agent's action may affect the world in a way irrelevant to her situation. This is possible if, for example, the agent's perceptual constraints do not allow her to perceive the effects of her actions. In these cases, the world may change, while the agent's situation remains the same. If we only modelled changes in an agent's situation, instead of the whole world as well, these effects would escape from view. Hence, it is important to keep track of changes in the world as well.

4.2.2. *World.* A world $\mathbf{W}$ is an algebraic structure $\mathbf{W} = \langle \mathrm{O}, \Omega\,\mathrm{Att}, \mathrm{E} \rangle$ such that:

- O is a sets of *objects*;
- $\Omega$ is an ordered set of sets of objects $\langle D_1, D_2, D_3, \ldots D_n \rangle$ (one for each agent in the system), called *domains*, such that $\cup Di \subseteq \mathrm{O}$, that is, every domain is a subset of O;
- Att is a set of *attributes*; and
- E, the *enviroment*, is a set of functions $f_i^n : (\mathrm{O})^{\mathrm{n}} \rightarrow \mathrm{Att}$.

4.2.3. *True fact and information availability*. Given an interpretation $\boldsymbol{I}$ of formal language $L$ on world $\mathbf{W}$, we say that the factual formula $A$ determines a *true fact* in $\mathbf{W}$ iff the interpretation of $A$ in $\mathbf{W}$ is true. For example, the factual formula $P_i^n(\mathrm{t}_1, \mathrm{t}_2, \ldots \mathrm{t}_n) = y_j$ is true in $\mathbf{W}$ iff $\boldsymbol{I}(P_i^n(\boldsymbol{I}(\mathrm{t}_1), \boldsymbol{I}(\mathrm{t}_2) \ldots \boldsymbol{I}(\mathrm{t}_n)) = \boldsymbol{I}(y_j)$, where $\boldsymbol{I}$ assigns objects in the world $o \in O$ to object terms (constant and variables) occurring on terms $\mathrm{t}_1, \mathrm{t}_2 \ldots \mathrm{t}_n$, functions in the environment $f_i^n: (\mathrm{O})^n \rightarrow \mathrm{Att}$ to predictive constants, $P_i^n$ and attributes in Att to informational constants $y_j$ (in co-ordination with the interpretation of $P_i^n$). The rest of the first order language is interpreted in the traditional, compositional Tarskian way. For example, a factual formula of the $\mathrm{A} \wedge \mathrm{B}$ form determines a true fact in world $\mathbf{W}$ if the interpretation of A and B determine a true fact in $\mathbf{W}$. In the same way, a factual formula in the $\mathrm{A} \vee \mathrm{B}$ form determines a true fact in world $\mathbf{W}$ if the interpretation of A, B or both, determines a true fact in $\mathbf{W}$, etc. Also, given a factual formula $A$, we say that sufficient information for determining the truth of $A$ in a world $\mathbf{W}$ is *available* for an agent $\alpha_\mathrm{i}$, if $A$ is true in $\mathbf{W}$ restricted to $\alpha_\mathrm{i}$'s domain $\mathrm{D_i}$. (As usual, the restriction is on the interpretation of the terms, including those with free variables).

4.2.4. *To follow an instruction*. Given an instruction $A = B \Rightarrow C$, where $B$ is a factual formula and $C$ a formula of action, we say that the world $\mathbf{W}_2$ is accessible from $\mathbf{W}_1$ in the interpretation $\boldsymbol{I}$ through the instruction $A$—written '$\mathbf{W}_1 \rightarrow_A \mathbf{W}_2$'—for an agent $\alpha_\mathrm{j}$ iff $B$ is false in $\mathbf{W}_1$ or the following three conditions are satisfied:

- the information necessary to determine the truth of $A$ is avaliable to $\alpha_\mathrm{j}$,
- the factual content of $C$ is true in $\mathbf{W}_2$, and
- being $\{t_{i1}, t_{i2}, \ldots t_{in}\} \rightarrow \{t_{i(n+1)}, t_{i(n+2)}, \ldots t_{i(n+k)}\}$ the object content of $C$, $\{\boldsymbol{I}(t_{i1}), \boldsymbol{I}(t_{i2}), \ldots \boldsymbol{I}(t_{in})\}$ $\mathrm{D_j}$ in $\mathbf{W_2} = \emptyset$ and $\{\boldsymbol{I}(t_{i(n+1)}), \boldsymbol{I}(t_{i(n+2)}), \ldots \boldsymbol{I}(t_{i(n+k)})\} \subseteq$ $\mathrm{D_j}$ in$\mathbf{W_2}$.

In other words, as long as to follow an instruction alters the state of things in the world, a world is accessible from another through an instruction iff following the instruction makes it evolve from one state to the other.

4.2.5. *To follow an instruction in MA systems*. One of the central characteristics of MA systems is their capacity to simultaneously contain several active agents following different instructions. However, this useful advantage of MA systems presents complications when its behaviour is logically modelled. It is not enough to model the changes in the world resulting from an agent's action. It is necessary for the simultaneous actions of different agents to be included in the model. Here is where the semantics of MA systems differs from traditional dynamic logic.

Given an agent $\alpha_\mathrm{i}$ whose set of instructions is $\mathrm{P_i}$, we say that the world $\mathbf{W}_2$ is accessible for á$_\mathrm{I}$ from $\mathbf{W}_1$ in the interpretation $\boldsymbol{I}$—written '$\mathbf{W}_1 \rightarrow \alpha_\mathrm{i}$ $\mathbf{W}_2$'—if $\mathrm{I}(\alpha_\mathrm{I}) =$ *inactive* is true in $\mathrm{M_1}$, that is to say, if the agent is inactive in $\mathrm{M_1}$, or $\mathbf{W}1 \rightarrow_p \mathbf{W}_2$ for every instruction $p$ in $\mathrm{P_i}$, that is to say, the world $\mathbf{W}_2$ is accessible from $\mathbf{W}_2$ according to every instruction in the agent's program.

Given a program $\boldsymbol{P_S} = \langle\langle \mathrm{P}_1, \mathrm{P}_2, \mathrm{P}_3, \ldots \mathrm{n}\rangle, A\rangle$, we say that the world $\mathbf{W}_2$ is accessible from $\mathbf{W}_1$ for the system $\boldsymbol{S}$ under the program $\boldsymbol{P_S}$ in the interpretation $\boldsymbol{I}$—written '$\mathbf{W}_1 \rightarrow_\mathrm{S} \mathbf{W}_2$'—iff $\mathbf{W}_1 \rightarrow \alpha_\mathrm{i}\, \mathbf{W}_2$ for every agent $\alpha_\mathrm{i}$ in $\boldsymbol{S}$. Notice that the algebraic frame used in our semantics has no correspondence with any of the traditional systems of modal logic (Chellas 1980, Boolos 1993). This is because each

program defines an accessibility relation between the possible worlds within the frame.

4.2.6. *To finish a program and model*. Given an interpretation $\boldsymbol{I}$, a program $\boldsymbol{P_S}$ and a collection of worlds $\mathbf{M}$, we say that the program $\boldsymbol{P_S}$ *finishes in a world* $\mathbf{W} \in \mathbf{M}$ iff no other world in $\mathbf{M}$ is accessible from $\mathbf{W}$ for system $\boldsymbol{S}$ under program $\boldsymbol{P_S}$. Also, an algebraic structure of worlds $\langle \mathbf{M}, \rightarrow_M \rangle$ is a *model* of the system's behaviour iff:

(1) accessibility relation $\rightarrow_M$ is defined on $\mathbf{M}$ so that $\rightarrow_M = \rightarrow_S$,
(2) there is one and only one original world $\mathbf{W}_0$ not accessible from any other world in $\mathbf{M}$, and
(3) the program finishes in one and only one world in $\mathbf{M}$.

Notice that here we are using the word 'model' in its semantic form, that is, as true interpretation. Given an interpretation $\mathbf{I}$ of a program $\mathbf{P}_S$, our model includes all the possible steps that an agent can take when executing the program, from an original state of the world (represented by $\mathbf{W}_0$) until the moment that it stops (represented by the world in which the program ends). The intuition behind this kind of semantic modelling is that the accessibility relation $\rightarrow_M$ represents the different courses of action that the multi-agent system can take (remember that $\rightarrow_S$ is generated from the combination of the different partial relations of accessibility $\rightarrow \alpha_i$ correspondent to each one of the agents in the system).

4.2.7. *Success*. We say that the program $\boldsymbol{P_S}$ is *successful* if its goal is true in every model of its behaviour, that is, in every true semantic interpretation of its program. Notice that, given that the goal formula is modal, it should not be evaluated in a particular world, but in the model as a whole. This is because the goal of a program may not always be described merely by how things are after successfully running the program (or at any other particular moment), but rather it may require evaluating the result of certain actions carried out during the development of the task.

This success can be *absolute*, if the program is successful for any collection of worlds, or *relative* to a given class of these worlds; commonly those that satisfy certain restrictions coded in a theory of the environment in which the agent is deployed. This last restriction accommodates those occasions in which the result of the analysis is not expected to be applied to any logically imaginable circumstance, but only to those in which it is reasonable to carry out the task. For example, in the case of the postman, it would be strange to wonder if the program works when there are some letters whose addresses do not correspond to any house, or some whose destination is materially inaccessible. In these cases, it is impossible for the program to be successful in an absolute sense, since not every letter can be successfully delivered. However, if it works perfectly in the *normal* cases, we may not want to say that the program just does not work. Hence, we say that it works, but constrained to *normal* circumstances. It is in these cases that the distinction between the absolute and the relative success of a program becomes important.

Once established this semantics, we have a clear, formal and computable approach to the success of cognitive task analysis. Furthermore, this semantics lays very close to the intuitive understanding of the cognitive work of agents, since it clearly illustrates the cognitive interaction between agent and world, starting from the

information obtained from its situation and the attributes of objects in the environment.

Next, we will illustrate the operation of the language with three behaviours: *battery*, *delivery* and *planning*, based on the analysis in section 3.

## 5. Formalization and verification of behaviours: *delivery*, *walk*, *battery* and *emergency*

This section illustrates one possible application of the logic developed in the previous section. It is applied in the verification of some of the results of the CTA analysis to the post delivery task presented in section 3. This section's goal is to show, by an example, how the syntax of our logic (section 4.1) allows for a detailed formalization of cognitive task analysis' results, while the semantic framework (section 4.2) allows us to verify the success of a behaviour, by modelling the universe (section 2.2) in which such behaviour takes place.

### 5.1. *World*

As it was established in sections 2.4.3 and 2.4.4, it is necessary to specify the kind of worlds where the multi-agent system postman will carry out its task. These worlds are structures of the form $\mathbf{W} = \langle \mathrm{O}, \Omega, \mathrm{Att}, \mathrm{E} \rangle$ such that:

- The set of *objects* O in the world is formed by:
  (1) The battery sensor $\{sensor\}$,
  (2) The letters $\mathrm{C} = \{c_1, c_2, \ldots c_k\}$,
  (3) The agents $\mathrm{S} = \{planner,\ optimizer,\ emergency,\ delivery,\ return,\ walk,\ battery\}$.
  (4) and the addresses stated by the delivery plan $Plan = \langle d_0, d_1, d_2, \ldots d_j \rangle$, where $d_0$ is the post office.
- The set of *domains* $\Omega$ contains for each agent $\alpha_\mathrm{i}$, a set of objects $\mathrm{D_i}$ called its *domain*
- The set of *attributes* Att includes:
  (1) The agent's possible locations D. In normal cases, D will include the destinations stated by the plan as well as the addresses written on the envelopes of the letters to deliver.
  (2) The attributes of being *active* or *inactive*, characteristic of the agents.
  (3) The possible values of reading of the battery sensor. To simplify the example we will only consider two of them: *charged battery* and *battery in need of recharge*, although this second is formally dispensable.
- The environment E contains the following functions:
  (1) Function *address*: $\mathrm{C} \rightarrow \mathrm{D}$ that assigns to each letter, its destination's address.
  (2) Function *next address nd*: Plan $\rightarrow$ Plan that assigns to each destination in the plan, the following destination.
  (3) Function *location loc*: $\mathrm{S} \rightarrow \mathrm{D}$ that assigns to agents their spatial location. In this case, we have the special restriction that all the agents that form the system move together, in such a way that the location of one will be the same for all.
  (4) Function *reading read*: $\{sensor\} \rightarrow \{charged\ battery,.\ battery\ in\ in\ need\ of\ recharge\}$ that assigns to the sensor its possible values, and

(5) Function *activity status*: $S \rightarrow \{active, inactive\}$ that assigns to each agent its condition of active or inactive at a given time.

### 5.2. *Syntax*

To obviate the formal language, we will use the objects' names as constants for the formalization of instructions. For example, since the agent *walk* is an object of the world, the symbol '*walk'* will be used as object constant in the syntax of the language. This is with the purpose of making more explicit and simple the connection between syntax and semantics.

### 5.3. *Agent's programs*

To illustrate our method we will only use four agents as examples:

Example 1: *delivery*. $\mathbf{P}_{delivery}$ has the following two instructions:

$$p_1 = \exists x\,(address(x) = loc(delivery)) \Rightarrow (status(walk) = inactive, \{x\}) \rightarrow \varnothing)$$

$$p_2 = \forall x(\neg(address(x) = loc(delivery)) \Rightarrow status(walk) = active$$

Let us analyse in detail what each one of these instructions says exactly. We begin with $p_1$. The factual formula ($\exists x$ *address*($x$) = *loc*(*delivery*)) expresses the fact that there is a letter $x$, in the agent's possession, whose address *address* is the current location *loc* of the agent *delivery*. As long as this factual formula appears to the left of the deontical arrow $\Rightarrow$, the fact there described corresponds to what should happen for the action in this first instruction to be followed. The action formula (*status*(*walk*) = *inactive*, $\{x\} \rightarrow \varnothing$) appearing to its right expresses the action to be carried out if such condition is satisfied. This last one has two sub-formulae: a factual and an objects-exchange. Its factual component is the formula *status*(*walk*) = *inactive*. This formula simply says that the agent *walk* must become inactive. Its component of changes of objects is the formula $\{x\} \rightarrow \varnothing$. Remember that in a formula of objects-exchange, the set to the left contains the objects that will stop to fall under the agent's power, while the one to the left represents the new objects over which the agent will obtain power. In this case, the set to the left only contains the object $x$, that is to say, the letter in whose address the agent *delivery* is at the moment. The set to the right is empty. This means that the agent drops the letter $x$ and does not pick up any other object at this time. Notice that this action formula contains a factual component as well as an exchange of objects one. This means that the action implied in this instruction involves the change of attributes of an object of the world (passing the agent *walk* of active to inactive), as well as leaving or picking up objects (the letter or letters to be dropped at that location). The first part is expressed in the factual component, while the second one is expressed in the objects-exchange component.

Once we have interpreted each component of the instruction it is easy to see what it means: if a letter $x$ on the agent's domain—that is, if it is still in her possession—is addressed *address* to the place where agent *delivery* is located *loc*, it should deactivate agent *walk* and drop letter $x$.

Let us pass now to analyse the second instruction. The factual formula $\forall x \neg(address(x) = loc(delivery)$ expresses the fact that some letter $x$ is not addressed *address* to the current location *loc* of the agent *delivery*, that is to say, that the system no longer needs to deliver any letter there. If this is so, what the

agent should do is expressed in the action formula *status(walk) = active*. This formula simply says that the agent *walk* should be activated. In consequence, this second instruction says that if the system does not need to deliver any letter in the place where the agent *delivery* is, it should continue walking. Notice that this instruction does not say that *if there are still letters to deliver*, the system should continue with delivery. For that it would be necessary to add a new condition $\exists x\,(\neg(address(x) = address(delivery))$ to this second instruction. However, this is the work of another agent, *return*, which is activated, and deactivates the agent *delivery,* when delivery is finished.

It is easy to see how this program fits the actions assigned to the agent in its behaviour scheme, such as this was presented in section 3:

*AnAgent'sActions:* to walk delivering mail.

The agent *delivery* achieves the action of walking by activating agent *walk*. When walking, the agent *walk* acts in a passive way regarding the goals imposed by the agent *delivery*. On the other hand, the *delivery* agent performs the action of delivering directly. To drop a letter is nothing more than to leave it in the place where the agent is. This kind of action is merely physical. Therefore, its formalization requires an instruction whose formula of action includes a change of objects. In this sense, to drop a letter is not more than to change the ontologic status of an object: the letter passes of being under the agent's possession to becoming a mere part of the environment.

Furthermore, this way of formalizing an agent's behaviour lets one clearly see what information is relevant for the task's performance. Here we have seen that the only source of information upon which agent *delivery* decides how to act is the address of the letters. In general, we can formulate the following formal definition of the concept of *relevant source of information*:

> Given a multiagent system $\boldsymbol{S}$, an object $o$ from the world $\boldsymbol{W}_S$ is a source of relevant information to an agent $\alpha$ in $\mathbf{S}$ if $o$ appears as constant of object in the factual formula antecedent of some instruction $p$ in $\mathbf{P}_\alpha$.

Consider now, other, simpler, agents.

Example 2: *battery*

$$\mathbf{P}_{battery} = \{read(sensor) = battery\ in\ in\ need\ of\ recharge\} \Rightarrow status(emergency) = active\}$$

Agent *battery*'s program is extremely simple, since its only action is activating the emergency agent when it reads in the sensor that the battery is in in need of recharge.

Example 3: *walk*

$$\mathbf{P}_{walk} = \{(loc(delivery) = loc(delivery)) \Rightarrow (loc(delivery) = nd(loc(delivery)))\}$$

The case of agent *walk* is especially interesting for the apparent simplicity of its program. From the point of view of the cognitive modelling of the system, what the agent *walk* does is very simple, not because the agent's internal operation is simple, but because it involves very little exchange of information with its environment and with the rest of the agents that conform the system. To move in space can be a very complicated task from the computational point of view, but cognitively, it is very easy to model: the only thing this agent does is to move

the system from one destination to another following the plan assigned to it by agent *planner*.

Example 4: *Emergency*

$\mathbf{P}_{emergency}$ contains the following instructions:

$$p_3 = (\neg(loc(delivery) = d_0)) \Rightarrow (status(delivery) = inactive) \wedge (status(return) = active)$$

$$p_4 = (loc(delivery) = d_0) \Rightarrow (read(sensor) = charged\ battery) \wedge (status(delivery) = active) \wedge (status(return) = inactive) \wedge (status(emergency) = inactive))$$

The first instruction says that if the location *loc* of agent *delivery* is not the post office $d_0$, it is necessary to deactivate agent *delivery* and activate agent *return*. In other words, if there is an emergency, it is necessary to suspend the delivery of the mail and to return to the postal office.

The second instruction says that if agent *delivery*'s location *loc* is the post office $d_0$, it is necessary to do three things: (1) ($read(sensor) = charged\ battery$) make *charged battery* the reading *read* of the *sensor*, (2) ($status(delivery) = active$) reactivate agent *delivery* and (3) ($status(return) = inactive) \wedge (status(emergency) = inactive)$) to deactivate agents *return* and *emergency*. This means that, when arriving at the postal office, agent *emergency* should charge the battery, renew delivery of the mail and cancel the emergency state.

### 5.4. *Verification of behaviour*

Once syntax and semantics are in place, the verification of an agent's behaviour consists simply in evaluating the program's success in all intended models. Verifying the program's success, in turn, does not require more than evaluating the program's goal formula as true in all acceptable models.

5.4.1. *Preliminary requirements*. To verify the behaviour of a system, it is necessary to have formalized the instructions of all agents first. However, this would make this illustrative section of the article too cumbersome. We can easily do with just the later examples and some intuitions on the operation of the rest of the agents to give a vision more or less schematic of how to semantically verify the behaviour of the system.

Before verification properly, it is necessary to do two things:

(1) As we had said in section 5, to evaluate the behaviour of a multi-agent system, we need to formalize, not only the instructions that conform each agent's behaviour, but also how we will evaluate the system's success. In other words, it is necessary to have a formula of the formal language that expresses the system's goal. In this case, the goal is very simple: to deliver the letters. The program is successful if, when finishing, the robot returns to the station and each letter is in the location to which is addressed. These two conditions define the behaviour's success.

Formalizing the first condition is very simple, although its modal expression is not directly obvious:

$$\Diamond\Box\,(loc(delivery) = d_0)$$

Semantically, this formula tells us that there must be a time in the development of the program when agent *delivery* arrives to $d_0$, that is to say, to the postal office; and it does not go anywhere from there.

The second condition is formulated in a similar way:

$$\forall c((\Diamond\Box loc(c) = address(c)))$$

What this formula tells us it is that, for every letter $c$ there must be a time when its location *loc* is the same as its address *address* and, from that moment on, it does not change location.

The goal formula for the program is, then:

$$A = \Diamond\Box(loc(delivery) = d_0) \wedge \forall c(\Diamond\Box(loc(c) = address(c)))$$

(2) It is clear that the success that interests us is not absolute, but relative, in such a way that, besides specifying the goal of the behaviour, it is also required to provide certain limitations on how the world must be for the task to be performed. These limitations can be expressed in the meta-language as formal constraints on the logical model.

Before anything else, we need the letters to be addresses to accessible locations. This condition is formalized the following way:

$$\forall c\, address(c) \in \mathrm{D}$$

We also need that, once the postman has picked them up at the station, letters do not move on their own, but only be moved by the *delivery* agent. This is formalized the following way:

$$\forall c\, loc(c) = loc(delivery)$$

After satisfying these two conditions, we can finally pass to the behaviour's verification, properly speaking.

5.4.2. *Formal verification*. We say that program $\boldsymbol{P_S}$ is *successful* if its goal is true in every model. In this case, this means that $A$ should be true for every model that satisfies $C_1$ and $C_2$. From a logical point of view, we could also say that $A$ should be a logical consequence of $C_1$ and $C_2$ under $\boldsymbol{P_S}$.

The formal test takes the form of a reduction. Suppose that $A$ is false in some model of $\boldsymbol{P_S}$. Then, at least one of it sub-formulae must be false as well. We begin considering the first one:

$$\Diamond\Box(loc(delivery) = d_0)$$

For this formula to be false, there should be a world $\mathbf{W}$ such that, in every world $\mathbf{W}'$ accessible from $\mathbf{W}$, $loc(delivery) \neq d_0$.

It is necessary now to revise the programs defining the system's accessibility relation in search of instructions capable of affecting the agent delivery's location. Clearly, the only agent with this capacity is *walk* whose program is:

$$\mathbf{P}_{walk} = \{\, (loc(delivery) = loc(delivery)) \Rightarrow (loc(delivery) = nd(loc(delivery))) \,\}$$

This is the time to pay closer attention to the activation and deactivation relations among agents of the system shown in figure 1, and whose semantics is defined by the direction of the arrows.

Given that agent *walk* is passive regarding other agents: *planner, delivery* and *return*, we should find in what situations any of these agents can activate or deactivate *walk*. What interests us at this time is when *walk* finishes, to see if indeed it finishes at the postal office. Therefore, we need to know if there is a world in which some agent deactivates *walk* in such a way that no other agent can reactivate it.

Only agent *delivery* has the ability to reactivate *walk*. This power comes into effect only when there are still letters to deliver, as it is stated in instruction $p_2$:

$$p_2 = (\forall x(\neg(address(x) = loc(delivery)) \Rightarrow status(walk) = active$$

As long as the set of letters is finite, it is clear that a time will arrive when *delivery* stops activating *walk*. The only thing we have to verify is that, in such case, *return* is activated.

In spite of the fact that we have not formalized it, it is clear that *return* should contain an instruction that not only deactivates *walk*, but also takes agent *delivery* back to the post office, when there are no more letters to deliver:

$$p_5 = \forall c\neg(address(c) = address(c)) \Rightarrow (loc(delivery) = d_0) \wedge status(walk) = inactive \in \mathbf{P}_{deliver}$$

It is clear that the only situation in which $\forall c\neg(address(c) = address(c))$ is a true fact is when $C = \varnothing$, that is to say, when there are no letters to deliver. In that case, *walk* is deactivated and returned to the postal office $d_0$. Neither should *return* have the capacity to reactivate *walk*. The agent *return*, on the other hand, can be activated by agent *delivery*, as shown in $p_3$:

$$p_3 = (\neg(loc(delivery) = d_0)) \Rightarrow ((status(delivery) = inactive) \wedge (status(return) = active))$$

What this instruction states is that, in the event of an emergency, all delivery of letters should be stopped, and agent *delivery* must activate *return*.

Finally, in spite of the fact that *walk* is passive regarding agent *planner*, the power of this last one over the first one is exercised through the delivery plan, not through the agent's activation/deactivation.

Once we have considered all possible cases where *delivery* finishes its task, we have found that in all of them, the agent has returned to the post office, which satisfies the first condition of the program's goal. Now we can pass to the second one:

$$\forall c(\Diamond\Box\,(loc(c) = address(c))$$

For this formula to be true, it is necessary that for every letter $c$, a world $\mathbf{W}$ exists, such that at any other world $\mathbf{W}'$ accessible from $\mathbf{W}$, letter $c$ is located *loc* at its destination *address*. To guarantee this, the program must satisfy two further conditions: (1) that the plan elaborated by agent *planner* includes the destinations of all the letters to deliver and (2) that agent *delivery* leaves them at their correct address. Since we have not formalized the instructions that constitute the *planner*'s program, we can assume that this first condition is satisfied. Regarding the second one, the test is relatively simple, given that the only agent with active access to the letters is agent *delivery*. Other agents receive information from the letters, but this agent is the only one capable to deliver them (hence, her name).

The relevant instruction is $p_1$:

$$p_1 = (\exists x(address(x) = loc(delivery)) \Rightarrow (status(walk) = inactive, \{x\} \rightarrow \varnothing)$$

Semantically, this instruction says that, given a world **W** such that *address(x) = loc(delivery)*, letter *x* should not exist in any world **W′** accessible from **W** via *delivery* (and that *walk* should be deactivated in **W′**, which guarantees that the agent won't move while delivering the letters). Assuming the good operation of *planner* and *walk*, we can guarantee that for any letter *c* there is a world $\mathbf{W_c}$ in the model such that *address(x) = loc(delivery)* is a true fact in $\mathbf{W_c}$. In other words, if *walk* and *planner* function properly, *delivery* should pass by all the addresses of all the letters.

Therefore, for all **W′** such that $\mathbf{W_c}$ *delivery* **W′**, *c* is not in *delivery*'s domain of objects $D_{delivery}$ in **W′**. From this, it vacuously follows that (*address*(*c*) = *loc*(*delivery*)) is true in **W′**, which guarantees that $\forall c(\Diamond\Box(loc(c) = address(c)))$.

Since we have proven the satisfaction of the two component sub-formulae of *A*, we can affirm that the goal is true in every model of $\boldsymbol{P_s}$, that is to say that the system is successful relative to conditions $C_1$ and $C_2$.

## 6. Conclusion

The previous logical system is a powerful tool for the formalization and verification of the behaviour of multi-agent systems based on CTA. Its syntax allows for a detailed formalization of cognitive task analysis' results, while its semantic framework allows for a simple mathematical way of verifying the success of a behaviour, by modelling the universe in which such behaviour takes place. While staying close to CTA's basic tenants, it exploits the strengths of modal (epistemic, deontic and dynamis) logics with a combination of possible world semantics and situation theory.

## Note

1. Another way to understand the main difference between traditional possible world semantics and situation theory is that the former was originally developed to account for the truth conditions of declarative sentences. Only later were they extended to cover also epistemic and informational phenomena. However, since there seems to always be some possible true knowledge beyond the reach of any particular agent in a given situation, it is clear for Barwise and his followers that different frameworks are required for each sort of phenomenae.

## References

Alvarado, M., and Sheremetov, L., 2001, Interaction Modal Logic for Multiagent Systems Based on BDI Architecture. *Memoria del 3er Encuentro Internacional de Ciencias de la Computación ENC01*, Sociedad Mexicana de Ciencias de la Computación, pp. 803–812.

Atkinson-Abutridy, J., and Carrasco, J., 1999, Un modelo evolutivo de redes de comportamiento para agentes autónomos que utilizan mecanismos de emergencia. En *Revista Iberoamericana de Inteligencia Artificial* (eds.) Asociación Española para la Inteligencia Artificial (AEPIA), No. 8, Otoño/99, pp. 59–67.

Baader, F., and Ohlbach, H., 1995, A multi-dimensional terminological knowledge representation language. *Journal of Applied Non-Classical Logics*, **5**, 153–197.

Barwise, J., 1988, Three views of common knowledge. *Theoretical Aspects of Reasoning About Knowledge II* (Pacific Grove, CA: Morgan Kauffmann Publishers).

Barwise, J., 1989, *The Situation in Logic* (Stanford, CA: Center for the Study of language and Information).

Barwise, J., and Allwein, G., (eds), 1993, *Working Papers on Diagrams and Logic* (Bloomington, IN: IULG Preprint Series No. IULG).

Barwise, J., and Perry, J., 1981, Situations and attitudes. *The Journal of Philosophy*, **78**(11): 668–691.

Barwise, J., and Perry, J., 1983, *Situations and Attitudes* (Cambridge, MA Bradford-MIT).

Barwise, J., and Seligman, J., 1997, *Information flow: the logic of distributed systems* (Cambridge: Cambridge University Press).

Beer, R. D., 1990, *Intelligence as Adaptive Behaviour: An Experiment in Computational Neuroethology* (Academic Press, Inc.).

Bertino, E., Guewrrini, G., and Rusca, L., 1999, Object evolution in object databases. In R. Pareschi and B. Fronhöfer (eds) *Dynamic Worlds*, Applied Logic Series 12 (Dordrecht: Kluwer), pp. 219–246.

Boolos, G., 1993, *The Logic of Provability* (Cambridge: Cambridge University Press).

Brooks, R., 1991a, Intelligence without representation. *Artificial Intelligence*, **47**: 139–159.

Brooks, R., 1991b, Intelligence without reason, Memo 1293, April, *MIT. Artificial Intelligence* Laboratory.

Cañas, J., and García-Alegre, M., 1999, Modulated agents for autonomous robot piloting. In *Memorias de la VIII Conferencia de la Asociación Española para la Inteligencia Artificial. CAEPIA'99*, Murcia, **1**: 98–106.

Castañeda, S., 1993, Procesos cognitivos y educación médica. *Serie* Seminarios, No.1. Facultad de Medicina – UNAM, México, DF.

Chellas, B., 1980, *Modal Logic: An Introduction* (Cambridge: Cambridge University Press).

Churchland, P. M., and Churchland, P. S., 1990, Could a Machine Think? *Scientific American*, **262**(January), 26–31.

Cooper R., Mukai, K., and Perry, J., (eds), 1990, *Situation Theory and Its Applications*, Volume 1, CSLI Lecture Notes Number 22, Center for the Study of Language and Information, Stanford, CA.

d'Inverno, M., and Luck, M., 2001, *Understanding Agent Systems* (Springer Verlag).

El Alami, M., de Arriaga, F., and Ugena, A., 1998, Multi-Agent simulation as an aid for decision making and learning. In *Proceedings of 2nd KFUPM Workshop on Information & Computer Science*, Dhaharam, Arabia Saudita, pp. 121–130.

Fischer, M. J., and Ladner E., 1977, Propositional modal logic of programs. *Conference Record of the Ninth Annual ACM Symposium on Theory of Computing*, 2–4 May, Boulder, CO, pp. 286–294.

Fischer M. J., and Ladner, E., 1979, Propositional dynamic logic of regular programs. *Journal of Computer and System Science*, **18**: 194–211.

Gabbay, D. M., Hogger, C. J., and Robinson, J. A., 1993, *Handbook of Logic in Artificial Intelligence and Logic Programming – Volume 1: Logic Foundations* (Oxford: Oxford University Press), pp. 175–240.

Galton, A., (ed.), 1987, *Temporal Logics and their Applications* (London: Academic Press).

Galton, A., 1995, Time and change for AI. In D. Gabbay, C. Hogger and J. Robinson (eds) *Handbook of Logic in Artificial Intelligence and Logic Programming*, Volume 4, Epistemic and Temporal Reasoning (Oxford: Oxford University Press), pp. 175–240.

García-Alegre, M., Bustos, P., and Guinea, P., 1995, Complex behaviour generation on autonomous robots: a case study. In *Proceedings of the IEEE International Conference on Systems, Man & Cybernetics*, Vancouver, Canada, **2**: 1729–1734.

Girard, J.-Y., 1987, Linear logic. *Theoretical Computer Science*, **50**(1): 1–101.

Gregory, R.L., 1987, *The Oxford Companion to the Mind* (Oxford: Oxford University Press).

Harel, D., 1984, Dynamic logic. In D. Gabbay and F. Guenther (eds) *Handbook of Philosophical Logic, volume 2, Extensions of Classical Logic* (Dordrecht: Reidel).

Hustadt, U., Dixon, C., Schmidt, R. A., Fisher, M., Meyer, J.-J., and van der Hoek, W., 2001, Reasoning about agents in the KARO framework. In C. Bettini and A. Montanari (eds) *Proceedings of the Eighth International Symposium on Temporal Representation and Reasoning (TIME'2001)*. IEEE Computer Society, pp. 206–213.

Jennings, N.R., 2000, On agent-based software engineering. *Artificial Intelligence*, **117**: 277–296.

Jung, C.G., 1998, On the role of computational models for specifying hybrid agents. In *Proceedings Cybernetics and Systems'98* (eds) Robert Trappl University of Vienna and Austrian Society for Cybernetics Studies, **2**: 749–754.

Kaelbling, L., 1987, An architecture for intelligent reactive systems. In M. P. Georgeff and A. L. Lansky (eds) *Reasoning about Actions and Plans* (Los Altos, CA: Morgan Kaufmann), pp. 395–410.

Laureano, A., 1998, Los sistemas reactivos: un nuevo acercamiento a la inteligencia artificial distribuida. Revista NOVATICA, No. 132: 51–55, Spain.

Laureano, A., 2000, Interacción Dinámica en Sistemas de Enseñanza Inteligentes. Doctoral thesis, Instituto de Investigaciones Biomédicas, Universidad Nacional Autónoma de México.

Laureano, A., and de Arriaga, F., 1998, Multi-agent architecture for intelligent tutoring systems. *Interactive Learning Environments*, **6**(3): 225–250.

Laureano, A., and de Arriaga, F., 1999, El análisis cognitivo de tareas una herramienta para modelar la conducta de los sistemas de enseñanza inteligentes. *Revista Latina de Pensamiento y Lenguaje,* **2B**: 315–335, Mexico.

Laureano, A., and de Arriaga, F., 2000, Reactive agent design for intelligent tutoring systems. *Cybernetics and Systems*, **31**(1): 1–47.

Laureano, A., de Arriaga, F., and García-Alegre, M., 2001, Cognitive task analysis: a proposal to model reactive behaviours. *Journal of Experimental & Theoretical Artificial Intelligence*, **13**: 227–239.

Luck, M., and d'Inverno, M., 1995, A formal framework for agency and autonomy. *First International Conference on Multi-Agent System* (AAAI Press. San Francisco, CA), pp. 254–260.

Manna, Z., and Pnueli, A., 1993, Models for reactivity. *Acta Informatica*, **30**(7), 609–678.

Maes, P., 1993, Situated agents can have goals. *Designing Autonomous Agents Theory and Practice from Biology to Engineering and Back* (The MIT Press Cambridge, MA & London), pp. 49–70.

Meyer J.-J.Ch., and van der Hoek, W., 1995, *Epistemic logic for computer science and artificial intelligence*. *Cambridge Tracts in Theoretical Computer Science* 41 (Cambridge: Cambridge University Press).

Mylopoulos, J., Brodie, M. L., and Schmidt, J. W., 1984, *On Conceptual Modelling, Perspectives from Artificial Intelligence, Databases, and Programming Languages*, Intervale Workshop 1982 (New York: Springer Verlag).

Ndumu, D., and Nwana, H., 1999, A perspective on software agents research. *The Knowledge Engineering Review*, **14**(2), 1–18.

Newell, A., 1981, The knowledge level. *AI Magazine*, **2**(2): 1–20.

Nwana, H., 1996, Software agents: an overview. The Knowledge Engineering Review, **11**(3): 205–244.

Parikh, R., 1981, Propositional dynamic logic of programs: a survey. In Erwin Engeleter (ed.) *Proceedings of the 1st Working on Logic of Programs,* LNCS 125, (New York: Springer-Verlag), pp. 102–144.

Pratt, V., 1976, Semantical considerations on Floyd-Hoare Logic. In *Proceedings of the 17$^{th}$ IEEE Symposium on Foundations of Computer Science*, pp. 109–121.

Reeding, R. E., 1992, A standard procedure for conducting cognitive task analysis. *ERIC. Documentation Reproduction Service*, No. DE 340-847.

Spivey, J. M., 1988, *Understanding Z: A Specification Language and its Formal Semantics* (Cambridge: Cambridge University Press).

Turing, A., 1950, Computing machinery and intelligence. *Mind*, **LIX**(236): 433.

Turing, A. M., 1947, Lecture to the London Mathematical Society. *The Collected Works of A. M. Turing*, *volume Mechanical Intelligence* (Amsterdam: D. C. Imce), pp. 97–106.

Van Benthem, 1983, *Modal Logic as a Theory of Information* Report LP-89-05, ITLI, University of Amsterdam.

Wolter, F., and Zakharyaschev, M., 1998, Satisfiability problem in description logics with modal operators. In *Proceedings KR-98* (Pacific Grove, CA: Morgan Kaufmann), pp. 512–513.