Adaptive behaviors for virtual entities in participatory virtual environments

Cédric Buche

▶ To cite this version:

Cédric Buche. Adaptive behaviors for virtual entities in participatory virtual environments. Artificial Intelligence [cs.AI]. Université de Bretagne occidentale - Brest, 2012. tel-00672518

HAL Id: tel-00672518

https://theses.hal.science/tel-00672518

Submitted on 21 Feb 2012

University of Western Brittany (france)

— Accreditation to Direct Research —

Field: Computer Science

Adaptive behaviors for virtual entities
in participatory virtual environments

Cédric Buche

Defense : February 10th, 2012

Reporters

Yves	Duthen	Professor IRIT	Toulouse
Jean-claude	Martin	Professor LIMSI	Paris 11
Patrick	Reignier	Professor LIG	Grenoble

Examiners

Daniel R.	Mestre	Research director CNRS	Marseille
Jacques	Tisseau	Professor UBO / ENIB	Brest
Pierre	De Loor	Professor UBO / ENIB	Brest

Adaptive behaviors for virtual entities in participatory virtual environments

Accreditation to Direct Research

Cédric Buche

— February 20, 2012 —

Laboratory of Science and Technology of Information, Communication and Knowledge (Lab-STICC) UMR 3192

CÉDRIC BUCHE
✉ : buche@enib.fr
🛈 : http://www.cerv.fr/~buche
✆ : +33 (0)2 98 05 89 66

Brest national engineering school (ENIB)

Technopôle Brest-Iroise
Parvis Blaise Pascal
C.S. 73862 F-29208 Brest Cedex
🛈 : http://www.enib.fr

European center for virtual reality (CERV)

Technopôle Brest-Iroise
25, rue Claude Chappe
BP 38 F-29280 Plouzané
🛈 : http://www.cerv.fr

Contents

Contents of Chapter 1

Chapter 1

Introduction

THE present studies take place in the fields of virtual reality, artificial intelligence, knowledge representation and agent simulation. I am a member of the IHSEV team which belongs to the Laboratory of Science and Technology of Information, Communication and Knowledge (LABoratoire en Sciences et Techniques de l'Information, de la Communication et de la Connaissance - Lab-STICC, CNRS, UMR 3192). I conduct my research at the European Center for Virtual Reality (Centre Européen de Réalité Virtuelle - CERV).

1.1   Scientific view point

The CERV is a center of excellence in virtual reality with a European calling. The systems we aim to model are increasingly complex. This complexity is essentially due to the diversity of the components, structures and interactions brought into play. A complex system is thus an environment which is open (components appear/disappear dynamically) and heterogeneous (varied behaviors and morphologies). Virtual reality applications fully involve the end user in the simulation, which is closely related to the participatory design approach [Tisseau, 2001a].

Studies in virtual reality mainly rely on sensori-motor immersion of the human user within virtual universes [Fuchs et al., 2001a]. These virtual worlds offer the user the sensation of being within the environment and give him the opportunity to act. To be complete, "something must happen", and not only in terms of result of user's actions. The entities that populate virtual worlds must behave autonomously [Tisseau and Harrouet, 2003a]. This raises the following question: how can an entity be equipped with autonomous behavior in a complex virtual environment in which human participates?

Symbolic artificial intelligence techniques have been applied to define these behaviors. However, these techniques have limitations as they are mainly based on predetermined rules of behavior chosen by the designer. Despite this fact, in complex (open simulation, heterogeneous and participatory) virtual worlds, entities may have unpredictable behavior (behavioral variability of autonomous entities, free will of human users), thus creating new situations. When faced with situations unforeseen by the programmer, entities may display unsuitable behaviors. Therefore, the methodologies derived from adaptive artificial systems may contribute to overcoming these limitations. The present study focuses on the theme of adapting the behavior of autonomous entities in participatory virtual environments. Adapting their behavior means making changes in order to adjust to their environment. The aim of such adaptation is to make the behavior of virtual entities as believable as possible (*i.e.* similar to human behavior). For this purpose, we consider that entities should *learn* through experience; they must *anticipate* the behavior of others and the potential impact on the environment, and they must also use the *presence of the human* user in the virtual world to their advantage to adapt their behavior. Imagine a virtual world where, like humans, each entity would have its own behavior which would evolve automatically throughout the simulation. This is the aim of the research presented here.

1.2 Research program

The following sections describe our scientific approach (section 1.2.1) and our application fields (section 1.2.2). My research activities are then described (section 1.2.3). Finally, we position these research activities in terms of our scientific approach and our application fields (section 1.2.4).

1.2.1 Scientific approach

Our work examines the adaptation of autonomous behaviors for entities in participatory virtual environments. To address this issue, we will focus our research on three of the target behavioral properties when modeling virtual entities:

1. **the entity must learn by doing (learning)**;

2. **the entity is equipped with an autonomous simulated world to predict the behavior of "its" world (anticipating)**;

3. **the entity considers human participants as privileged actors to guide the adaptation of its behavior (human "in the loop")**.

My research addresses one or more of these three points. We will now explain these three behavioral properties.

1.2.1 - A The entity must learn by doing

To adapt, *i.e.* to change its behavior, the virtual entity must learn. Conventionally, machine learning techniques can be either supervised or free [Cornuéjols et al., 2002]. In supervised mode, an expert (supervisor) takes charge of the learning process (select examples, assess the responses). In free mode, the world evolves regardless of the learning entity. This method is consistent with the need for autonomy, which we want to guarantee. In addition, the virtual entity must be able to adapt itself in real-time. Our learning process will be unsupervised and performed in real-time. The entity continues to evolve in the virtual world and, in parallel, it modifies its behavior: the entity *learns by doing* (see figure 1.1).
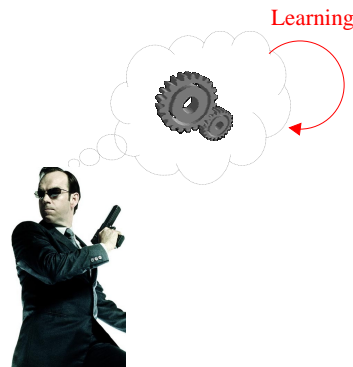
FIGURE 1.1 – Behavioral property 1: the entity learns by doing

1.2.1 - B The entity has its own simulated world

In order to adapt its behavior and to be believable, the entity must be equipped with an ability to anticipate. For this purpose, we propose a novel approach where the entity possesses an autonomous world of simulation within the simulation (see figure 1.2): it can simulate itself (using its own model of behavior) and simulate its environment (using its representation of other entities). The entity thus has the ability to anticipate using internal simulations, which would be extremely difficult using formal proofs methods in complex environments [Tisseau, 2004b].
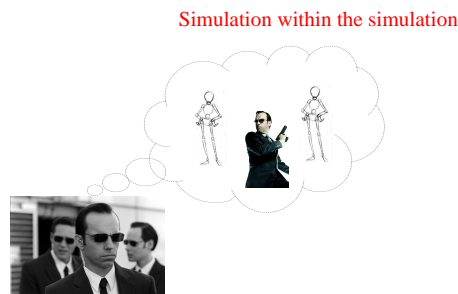
FIGURE 1.2 – Behavioral property 2: the entity simulates itself and its environment

This internal simulation ability, especially for movement, is now well established in neurophysiology [Brunia, 1999]. An entity performs these sensori-motor predictions, not by logical

reasoning or abstract symbols representative of the real world, but by biological simulation. *It is as if* the individual was really acting [Berthoz, 1997]. In the experiment called "The Waiter", [Hugon et al., 1982, Dufossé et al., 1985], a waiter is equipped with a tray on which there is a jug. The waiter takes the jug with his other hand. He then imagines the consequences of his movement (*i.e.* movement of the tray), and anticipates this effect by counteracting the weight discrepency with his carrying arm [Berthoz, 1997].

1.2.1 - C The entity considers the human as a privileged actor to guide the adaptation of its behavior

Humans *participate in our virtual worlds*. Human users can, according to a substitution principle, take control of an entity and identify himself as that entity. The controlled entity can then learn behaviors that are better adapted to its environment from the human operator (see figure 1.3). It is this type of learning, by example [Del Bimbo and Vicario, 1995] or by imitation [Meltzoff, 1995, Gallese, 2000], that we will use for modeling adaptive behaviors.

FIGURE 1.3 – Behavioral property 3: the entity considers the human as a key actor (picture from Microsoft's Project Natal [2])

1.2.2 Application fields

To better understand our problems on a fundamental level, our work is applied to specific fields:

▷ Video games.
  These applications are based on large-scale virtual environments in which real human actors are integrated and thus interact with the autonomous artificial entities be they human or animal, realistic or imaginary.

▷ Interactive art.
  These applications consist of artistic projects incorporating real-time interactions between both real and virtual artists. In this way, we can study the behavioral realism of the entities and their evolution.

---

[2] Project Natal: http://www.youtube.com/watch?v=g_txF7iETX0

▷ Virtual learning environments for humans.
These applications focus on interactive environments for human learning in the context of virtual reality. They allow the integration of adaptive Intelligent Tutoring Systems (ITS) to facilitate learning.

We also use "toy" environments to carry out initial tests on our models (simulated sheep and sheepdogs, mazes, scheduling software).

1.2.3 Research activities

My research activities have resulted in five major proposals.

1.2.3 - A Chameleon

This project is devoted to the thesis of Fabien Tence. This work is the subject of a partnership with the society Virtualys. It focuses on the acquisition of behavior (property 1) by imitation of human behavior (property 3) in virtual environments such as video games. The principle of learning is based on the fact that a user participates in the simulations, according to a principle of substitution, taking control of an entity and identifying with it. The controlled entity can then learn behaviors that are better adapted to its environment from the human. Unlike conventional approaches, the learning is not guided by criteria related to the performance of the entity (such as scores). It is conditioned by its believability in approaching "human standards". This project is applied to video games that seek to immerse players in simulations that want aim to be as real as possible, through simulated complex environments. Researchers can thus overcome some technical difficulties (rendering, physics, management, network, etc..) and focus on the implementation of behavior. In addition, video games are intented for humans, they offer a real challenge for the entities in terms of behavioral believability. Also, they provide a community of experts in these environments in the presence of human players, allowing to obtain critical relevant on the entity believability.

> **Publications:** [Tence et al., 2010b, Tence et al., 2010a, Tence and Buche, 2008]
> **Supervision:** F. Tence (M2R), F. Tence (Thesis)

1.2.3 - B Jabu

This project is part of our work on integrating a proactive process by internal simulation for our virtual actors (property 2) in environments involving the human user (property 3). This principle is illustrated by the development of an artificial 3D juggler. For this application, the juggler can be characterized as proactive: it predicts the behavior of balls in the air and uses their predictions to coordinate its behavior in order to juggle. In addition, an interface allows a human actor to throw a ball to the virtual juggler.

> **Publication:** [Buche et al., 2011]
> **Supervision:** A. Jeannin-Girardon (M2R)

1.2.3 - C CAMUS

This project uses the three behavioral properties that we have established. The perceptual behavior of virtual actors must determine their responses according to external stimuli and also according to internal emotions. The aim is to describe such emotional behaviors using fuzzy cognitive maps where these internal states are explicitly represented. We detail how fuzzy cognitive maps allow the specification, control, internal simulation (property 2) and the dynamic adaptation of perceptual behavior (property 1). We illustrate our approach through an example depicting a shepherd, his herd and sheepdogs. Each entity, according to a principle of substitution, can be controlled by a human user (property 3).

> **Publications:** [Buche et al., 2010b, Tisseau et al., 2005b, Popovici et al., 2004a, Buche et al., 2002]
> **Collaboration:** M. POPOVICI (Virtual and Augmented Reality Research Laboratory, Roumanie)

1.2.3 - D GEMEAU

This project focuses on the realization of a model for learning behaviors online (property 1) by classifiers systems. These are powerful tools for learning interactions between entities and their environments. However, there are many kinds of classifiers systems which differ in terms of numerous subtle technical features. This project analyzes the main kinds of classifiers systems in order to suggest a generic model common to all of them. The implementation of this model is flexible enough to be used for different problems since it provides an interface between the environment and the system. It can be used to quickly test many types of classifiers systems, to make different conceptual assumptions, and to obtain interesting comparative results. It has been used on different problems as labyrinth type applications and management schedule software. This model is also used for the PEGASE project.

> **Publications:** [Buche and De Loor, 2010, Buche et al., 2006d, Buche et al., 2006c]
> **Supervision:** E. CREAC'H (M2R)

1.2.3 - E PEGASE

This project is devoted to the thesis of FRÉDÉRIC LE CORRE, funded by the society DIAGNOSTICA STAGO. It contributes to our work on learning behaviors for our virtual entities (property 1) involving the human user (property 3) within the field of virtual environments for training. We incorporate an adaptive and generic intelligent tutoring system (ITS) within a virtual environment to provide educational assistance to the learner and pedagogical assistance to the teacher. Our study highlights the need for abstract representation, which must be independent of the task at hand, modifiable for pedagogical decision-making, and connected to the representation of a 3D universe. Our proposal is a multi-agent system for analyzing the action carried out by the learner using an informed virtual environment. The system highlights a set of information, called the pedagogical situation, considered relevant for making pedagogical decisions. Our study then focuses on a pedagogical agent. It suggests that the trainer should be assisted by the pedagogical situation. Abstraction provides concrete assistance linked to the field, the exercise, and the virtual environment. The behavioral

model of the pedagogical agent is based on a hierarchical classifiers system. Thanks to this model, the agent adapts to the trainer-learner pair, modifying its pedagogical behavior by means of an artificial learning mechanism (property 1) based on reinforcement provided by the trainer (property 3).

> **Publications:** [Querrec et al., 2011, Buche and Querrec, 2011, Buche et al., 2010a]
> **Supervisions:** G. Faudet (M2R), T.H. Trinh (M2R), Y. Cardin (M2R), F. Le Corre (Thesis)

1.2.4 Positioning projects

The studies presented in this paper have helped to address our problem based on three behavioral properties from our scientific perspective. For each project, behavioral properties have held precedent over other properties (see table 1.1). Similarly, each project is applied in several fields (see table 1.2).

Project / Behavioral properties	Jabu	Camus	Chameleon	Gemeau	Pegase
① Learning by doing		✓	✓	✓	✓
② Simulation within the simulation	✓	✓			
③ Human model for learning	✓	✓	✓		✓

TABLE 1.1 – Projects positioned according to our behavioral properties

Project / Application	Jabu	Camus	Chameleon	Gemeau	Pegase
Video games			✓		
Interactive art	✓				
Virtual learning environment				✓	
Toy examples		✓		✓	✓

TABLE 1.2 – Applications positioned according to our projects

1.3 Manuscript organization

The remainder of this paper is organized as follows:
Chapter 2 describes the Chameleon project.
Chapter 3 presents the Jabu project.
Chapter 4 describes the Camus project.
Chapter 5 is devoted to the Gemeau project.
Chapter 6 presents the Pegase project.
Finally, in chapter 7, we review our initial research and outline the future direction of research.

Contents of Chapter 2

Chapter 2

CHAMELEON — a bayesian model for believable behaviors in video game

In some video games, humans and computer programs can play together each one controlling a virtual humanoid. These computer programs usually aim at replacing missing human players but can be spotted as being artificial by players. Our objective is to find a method to create programs which behavior cannot be told apart from players when being observed playing the game. We call this kind of behavior a believable behavior. To achieve this goal, we choose a model developed by Le Hy to generate the behaviors by imitation. The model uses probability distributions to find which decision to choose depending on the sensors. Then actions are chosen depending on the sensors and the decision. The core idea of the model is promising but we propose to enhance the expressiveness of the model and the associated learning algorithm. We first revamp the organization of the sensors and motors by semantic refinement and add a focus mechanism in order to improve the believability. To achieve believability, we integrate an algorithm to learn the topology of the environment. Then, we revamp the learning algorithm to be able to learn much more parameters and with greater precision at the cost of its time of convergence.

2.1   Introduction

To make human user feel like he/she is in the simulation, two criteria have been defined in academic research: *immersion* and *presence*. According to Slater, immersion is an objective criterion which depends on the hardware and software [Slater et al., 1995]. It includes criteria based on virtual sensory information's types, variety, richness, direction and in which extend they override real ones. For example, force feedback and motion sensing controllers, surround sound and high dynamic range rendering can improve the immersion. Presence, also known as telepresence [Steuer, 1992], is a more subjective criterion. It is defined as the psychological sense of "being there" in the environment. As stated in [Slater et al., 1995], presence partly depends on the match between sensory data and internal representation. This match

expresses the fact that we try to use world models to better understand what we perceive and to be able to anticipate [Held and Durlach, 1991]. This idea is close to what is called *believability* in the arts. Indeed, we can believe in fictional objects, places, characters and story only if they mostly fit in our models.

As there are many ways to enhance believability, we choose to focus on believable virtual characters, also known as believable agents. The reason why we make this choice is because characters have often a major role in the believability of book and movie stories. However, unlike book and movie characters, agents should be able to cope with a wide range of possible situations without anyone to tell them what to do. Instead of defining manually these behavior, we propose that our entities will be able to learn, for them to be autonomous and have believable behaviors. This learning will be unsupervised and online: the entity will learn while it acts. This will both remove the burden of parametrizing the models controlling the characters and increase believability by having real-time evolution of the behavior.

To be able to achieve the best believability, we want the agents to do like human-controlled virtual characters. Indeed, there are no better example of what a believable behavior is than a human behavior itself. It is this kind of learning, by example [Del Bimbo and Vicario, 1995] or by imitation [Gorman and Humphrys, 2007, Bauckhage et al., 2007] we want to use to model believable and autonomous characters.

The way the characters act and learn depends heavily on the kind of virtual environment they are in. We share issues with the video games industry because the game designers want the players to be immersed in the simulation too. They are trying to be as close as possible from reality, making rich and complex environments. Researchers can avoid some technical difficulties (rendering, physics, networking, etc.) by using such games. They can then focus on the making of the entities they want to study [Cavazza et al., 2003, Mac Namee, 2004]. Furthermore, video games being made for human beings, they offer a real challenge for the entities to be believable. An other advantage is that they offer experts of these environments, human players, which can give pertinent criticism on the entities' behaviors [Silverman et al., 2006].

In this chapter, we first define what are believable agents, we give an overview of the kind of models which drive entities' behaviors and we focus on a model from Le Hy [Le Hy et al., 2004]. After explaining the key features of the model we present how we implemented it to find out how it can be improved (section 2.2). Then we propose some modifications in order to make the virtual character more believable and able of learning almost all the parameters of the model (section 2.3). Results are show in section 2.4. To conclude, we give explanations about how we want to evaluate the believability of our model (section 2.5).

2.2 Believable behaviors in video games

2.2.1 Believability

2.2.1 - A Definition of believability

The notion of believability is highly complex and subjective. To define and understand this concept, we must look at its meaning in the arts where it is a factor of *suspension of disbelief* [Bates, 1992a]. According to Thomas and Johnston, two core animators of Disney, believable characters' goal is to provide the "illusion of life" [Thomas and Johnston, 1981]. Reidl's definition is more precise: "Character believability refers to the numerous elements that allow a character to achieve the 'illusion of life', including but not limited to personality, emotion, intentionality, and physiology and physiological movement" [Riedl and Young, 2005, page 2]. Loyall tries to be more objective saying that such a character "provides a convincing portrayal of the personality they [the spectators] expect or come to expect" [Loyall, 1997b, page 1]. This definition is quite close to one factor of the presence, the match between players' world model and sensory data.

If we want to apply the believability definition for video games, things become even more complex. Unlike classic arts, players can be embodied in a game by the mean of virtual bodies and can interact. The believability question is now: does a believable character have to give the illusion of life or have to give the illusion that they are controlled by a player? [Livingstone, 2006]. There can be very important differences as even if the video game depicts the real world, all is virtual and players know that their acts have no real consequence.

In this research work, we consider only believable as *giving the illusion of being controlled by a player*. Now that we have defined believability, we have to find how to improve it and measure the improvement.

2.2.1 - B Believability criteria

As believability is a broad concept, we need to find more precise criteria to break this concept down. According to the litterature, we listed criteria which were reported to have an impact on believability. The requirements for believability are listed in table 2.1.

First the agent must react to the environment and the other players in a coherent way. This reaction must simulate a reaction time similar to human reaction time. The agent must also avoid repetitiveness, both in the actions and in the behavior. It is also necessary that the intention of the agent can be easily understood by the human players. Contrary to what is done in most video games, the perception abilities of the agent must be similar to those of a player. The agent should be able to handle the flow of time, remembering information from the past and thinking ahead, making plans. Finally the agent has to be able to evolve, changing its behavior for a more efficient and believable one. This evolution must be fast enough for the players to notice it, making them feel they play against an evolved being.

Believability requirement	Summary of the requirement	References
[B1: Reaction]	React to the players and changes in the environment	[Livingstone, 2006, Wetzel, 2004]
[B2: Reaction time]	Simulate a human-like reaction time	[Laird and Duchi, 2000, Livingstone, 2006]
[B3: Variability]	Have some variability in the actions	[Laird and Duchi, 2000, Livingstone, 2006] and [Loyall, 1997b, page 18]
[B4: Unpredictability]	Surprise the players with unpredictable behavior	[Bryant and Miikkulainen, 2006, Isla, 2005]
[B5: Understandable]	Have a understandable behavior	[Pinchbeck, 2008, Isla, 2005]
[B6: Perception]	Have human-like perception	[Cass, 2002, Mac Namee, 2004, page 34]
[B7: Planning]	Plan actions in the future to avoid mistakes	[Livingstone, 2006]
[B8: Memory]	Memorize information	[Loyall, 1997b, page 22]
[B9: Evolution]	Evolve to avoid repeating mistakes	[Thurau et al., 2005, Gorman and Humphrys, 2007]
[B10: Fast Evolution]	Evolve fast enough for the players to see it	

TABLE 2.1 – List of the requirements for an character to be believable.

2.2.2   Behavior models for believable agents

2.2.2 - A   Models criteria

As there are very few evaluations of the believability of behavior models, we will try to find the models which fulfil most of the requirements for believability. Because of the context of this study, we will only look at models for embodied agents in the following study. Those models can handle interactions with virtual environments and avatars. Therefore, they all fulfil the requirement [B1: Reaction]. According to the criteria we listed in section 2.2.1 - B, we list requirements for the model itself in the table 2.2.

2.2.2 - B   Behavior model choice

In order to fulfil these requirements, we studied the existing behavior models developed both in the research and the industry. We grouped behavior models into four types: connectionist models, state transition systems (*FSM*), production systems and probabilistic models. Connectionist models are very good at learning by usually have problems to handle memory and

Model requirements	Summary of the requirement
[M1: Variability]	Model variations in the actions and behaviors [B3: Variability] and [B4: Unpredictability]
[M2: Learning]	Is compatible with learning algorithms [B9: Evolution]
[M3: White box]	Can be modified and parametrized manually [Isla, 2005] to make the agent overdo [B5: Understandable]
[M4: Exaggeration]	Generate exaggerated behaviors so that players can easily understand the agent's objectives [B5: Understandable]
[M5: Planning]	Elaborate plans to avoid doing easily avoidable mistakes [B7: Planning]
[M6: Reaction time]	Simulate reaction time [B2: Reaction time]
[M7: Memory]	Model memory [B8: Memory]

TABLE 2.2 – Requirements for the models to make agents express believable behaviors.

planning. State transition systems can be easily understood and modified but may are not very well adapted to learning and planning. Production systems are quite good for learning, planning and memory but make the agent act in a predictable manner. Finally, probabilistic models are good at variability, learning and memory but may show problems with planning (see table 2.3).

	Connectionist	State transition	Production	Probabilistic
[M1: Variability]	✗	✗	✗	✓
[M2: Learning]	✓	✗	∼	✓
[M3: White box]	✗	✓	✓	✓
[M4: Exaggeration]	✗	✓	∼	✓
[M5: Planning]	✗	✗	✓	∼
[M6: Reaction time]	✓	∼	✓	✓
[M7: Memory]	∼	∼	✓	✓

TABLE 2.3 – Summary of the characteristics of models for the control of believable agents.

As one of the requirements is that the model is able to evolve, we had to find adapted learning algorithms. In order to achieve behavior believability, the best method we found is imitation learning: the agent learns its behavior using observations of one or several players. According to our definition of believability, it is the best way for the agent to look like players. Indeed, the goal of imitation learning is to make the agent acts as human players. This learning method is also much faster than trial and error.

With these studies in mind we found out that the behavior model developed by Le Hy in his thesis [Le Hy, 2007, in French] answers to most of the requirements. Le Hy's study presents a probabilistic model based on an input-output hidden Markov model (*IOHMM*): a hidden state is chosen according to inputs and the previous hidden state, and outputs are chosen according to inputs and the current hidden state. In Le Hy's behavior model, hidden states are decisions, inputs are stimuli and outputs are actions. Several learning algorithms have been developed for this model. The best combination is a Laplace's rule for the learning of the action distributions and an expectation–maximization (*EM*) for the learning of the Markovian distributions.

2.2.3   Le Hy' model

2.2.3 - A   Principle

In Le Hy's model, the agent has sensors named $S = (S_0, .., S_n)$. They give information on internal and environment's state like for instance the character's inventory and the position of another character. Agent's movements are driven by motors named $M = (M_0, ..., M_p)$ which can be rotation, jump commands and so on. Both sensors and motors take discrete values. In order to simulate the character's behavior, the notion of decision has been introduced, the associated variable is named D and may have different values like searching for an object or fleeing.

The value of D^t, where t is the time, is chosen according to the value of the sensors and to the previous decision, following the probability distribution $P(D^t|S^tD^{t-1})$. As S is the conjunction of n variables, Le Hy introduces the notion of inverse programming (IP) to reduce the complexity: $P(D^t|S^t)$ is computed using $P(S_i^t|D^t)$ (and not $P(D^t|S_i^t)$!) as they are assumed to be independent, which is a strong hypothesis.

Once the value of D^t is chosen randomly following the distribution $P(D^t|S^tD^{t-1})$, the model must decide which motor command should be activated. The value of each motor command is chosen following the distribution $P(M_i^t|S^tD^t)$, the motor model. Again, to reduce the complexity, Le Hy introduce the notion of fusion by enhanced coherence (FEC). Each command is computed separately then they are combined using the formula $P(M_i^t|S^tD^t) = \frac{1}{Z}\prod_j P(M_i^t|S_j^tD^t)$ where $1/Z$ is a normalization factor.

Thus the model, which can be categorized as an $IOHMM$, is composed of three types of parameters whose relations are summarized in figure 2.1: $P(D^t|D^{t-1})$, $P(S_i^t|D^t)$ and $P(M_i^t|S_j^tD^t)$
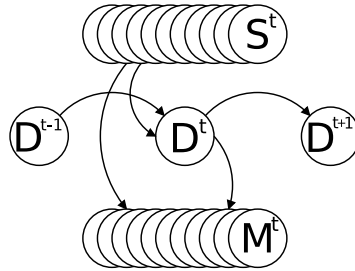
FIGURE 2.1 – Summary of the influences between model's variables [Le Hy et al., 2004].

Those parameters can be specified by hand or learned by imitation. Results seems to be better in term of believability and performance with learned parameters. The imitation is done by observation of the virtual representation of the player, his avatar also named the demonstrator. By monitoring at each time step the values for S and M for this demonstrator, it is possible to update the value of the parameters. The learning algorithm developed by Le Hy is based on the Incremental Baum-Welch (IBW) algorithm [Florez-Larrahondo, 2005] but only updates the decision model, the parameters $P(D^t|D^{t-1})$ and $P(S_i^t|D^t)$.

As believability is based on the feeling of an observer, we have to examine the behaviors produced by the model to see its real advantages and drawbacks (see section 2.2.3 - B).

2.2.3 - B   Implementation: BIBOT

To implement Le Hy's model, we used the game Unreal Tournament 2004, a newer version of the game Le Hy used for his model. It is a first person shooter game, in other words each player or agent controls a unique virtual character and sees through its eyes. The character can, non-thoroughly , grab items (weapons, . . .), move (walk, jump, . . .) and shoot with a weapon. Each character have an amount of hit points, also known as life points: each time an actor is hit by an enemy fire, a certain amount of hit points are subtracted to the current value. When hit points reaches zero, the character "dies" but can usually reappear at another place in the virtual environment. Although the concept can seem very basic, it can prove challenging for agents to resemble humans because it requires many abilities: fast reactions, accuracy, planning, adaptation to the environment and the other players, etc.

The model was built using the Pogamut framework [Burkert et al., 2007]. This one was developed in order to interact easily with the Gamebots interface that controls remote bots in Unreal Tournament 2004. The communication between the model and the environment is realised by messages over the network. The architecture is quite simple, we conceived our agent which lays on the definition of an agent provided by Pogamut (see Figure 2.2). That gives us access to the key components of an Unreal Tournament avatar which are the body and the memory. On one side, the body deals with all the motor instructions and on the other side the memory store all the perceptions the avatar can get during its evolution in the virtual environment.
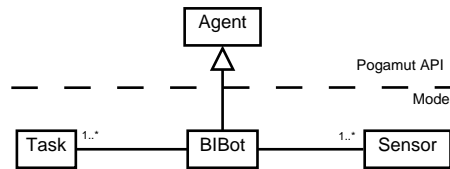
FIGURE 2.2 – Our agent inherit from Pogamut agent

In our model that we called BIBOT for Bayesian Inference based Bot, we decided to categorise these aspects in TASKS and SENSORS as suggested in [Le Hy et al., 2004]. We insisted on making it modular so the model can be extended by the addition of new tasks and/or sensors (see Figure 2.3).

To test the model, we used the probabilities tables values as specified in Ronan Le Hy's thesis (aggressive manual specification). When BIBOT is evolving by itself in the environment hence doesn't feel in danger, the tasks are switching between *Weapon Search* and *Explore* as we can see on Figure 2.4(a). That balance can be perturbed by the adding in the environment another player (human or not) which will be considered by BIBOT as a threat. Therefore its behavior change according to the modification of the surrounding environment (See Figure 2.4(b)).
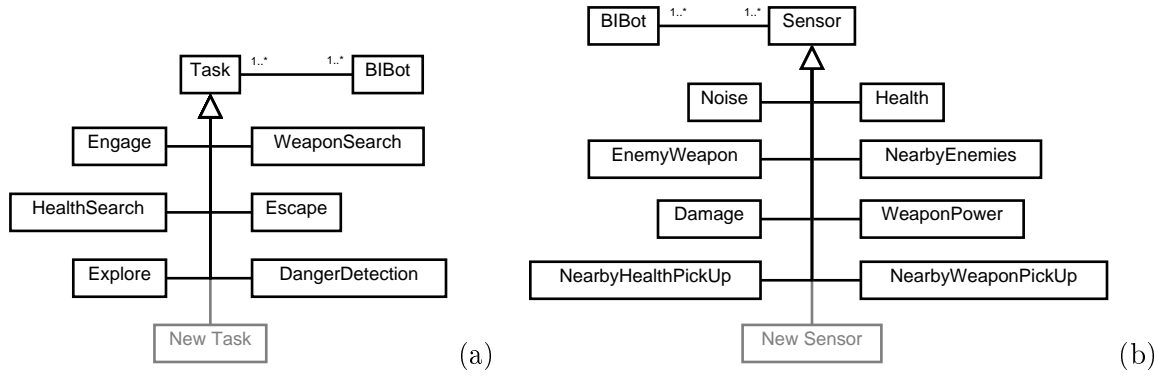
FIGURE 2.3 – Extendable Tasks (a) and extendable Sensors (b)
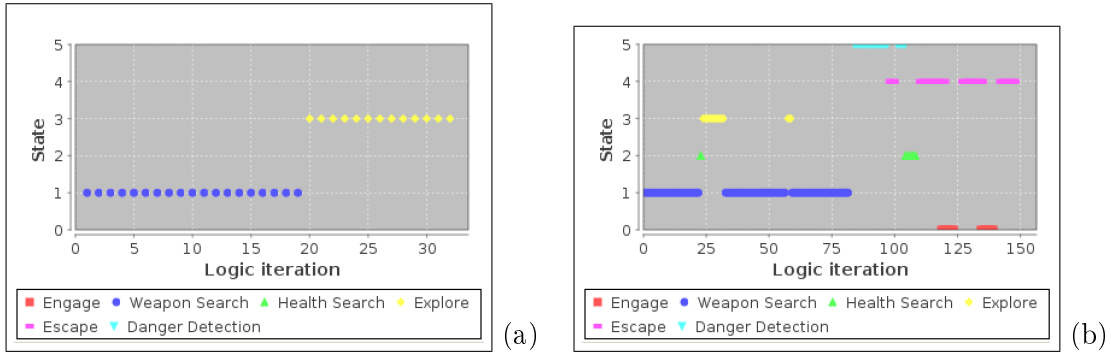
FIGURE 2.4 – Task sequencing before perturbation (a) and after perturbation (b)

2.2.3 - C   Limits of the model

BIBot allowed us to spot the strengths and weaknesses of both the architecture and the resulting behaviors of Le Hy's model. As the architecture of the model reminds of a *FSM*, it is easy to understand and to adjust the parameters. The model is modular, allowing the programmer to add or remove sensors, decisions and actions without modifying the code too much.

However, the behaviors produced by the model in the game can easily be spotted as artificial by casual and regular players. That is the first of the two main reasons we preferred not to run a complete experiment of the believability of the model. The second reason is that such experiment is very complex and time-consuming [Tence et al., 2010a]. However, in the future, we will assess the believability of both Le Hy's model and our model to confirm our first impressions.

We conclude by pointing out several problems (see table 2.4) in both its design and results: the independence hypothesis between actions, the *FEC*, the *IP* and the approximations made in the *IBW* make the agent unable of having and learning complex Behaviors. The model has also difficulties in adapting to new environments.

Problems	Summary of the problem
[P1: Navigation]	The agent has problem navigating in environments.
[P2: Sensors]	The sensors are not well organized.
[P3: Expressiveness]	The *FEC* does not provide enough expressiveness for the agent to be believable.
[P4: Scaling]	The *IP* have problem handling many sensors.
[P5: Readability]	The *IP* is not easy to read for novices.
[P6: Learning]	The learning algorithm use strong hypothesis which may hinder its capabilities.

TABLE 2.4 – List of the noticed limitations with Le Hy's model.

2.3 CHAMELEON

This section describes four enhancements of Le Hy's work to achieve more believable behaviors. In section 2.3.1, we split the sensors into two types representing two granularities of information: high-level stimuli and low-level stimuli to clarify [P2: Sensors]. We also defined two kinds of actions each one associated to a kind of stimuli. Then we propose an attention selection mechanism where the agent selects one high-level stimulus and one low-level which answers to the problems [P4: Scaling] and [P5: Readability]. This mechanism makes the model more flexible and allows the model to express more complex behaviors, solving [P3: Expressiveness]. In section 2.3.2, we propose to use a modification of the *GNG* algorithm to learn by imitation information about the layout of the environment giving an answer to [P1: Navigation]. Finally, in section 2.3.3, we propose a revamped imitation learning algorithm to learn almost all the model parameters resolving the problem [P6: Learning].

2.3.1 Improvements on Le Hy's model

2.3.1 - A Semantic refinement

Principle

The problem [P2: Sensors] is related to the independence between the stimuli and the other random variables in the model. We found that when implementing the model, decision and actions were obviously independent from some stimuli. Therefore, it is possible to alleviate the work of the learning algorithm by specifying from the beginning the independences.

The goal of the semantic refinement is to reduce the number of parameters by defining *a priori* some independence between random variables. By reducing the number of parameters and giving the model some knowledge, the learning should be faster without reducing the expressiveness of the model.

The principle of the proposition (see figure 2.5) is the following: instead of considering all the stimuli for each choice (decision and actions), each choice uses stimuli with different

level of granularity. Global and spatially inaccurate stimuli, called high-level stimuli (random variables H_i^t), are used for the choice of the decisions and also for actions which only involve the agent. Spatially accurate stimuli, called low-level stimuli (random variables L_j^t), are used to perform actions which aim at interaction with the environment. We also defined two kinds of action, reflexive, R_i^t, and external actions, E_i^t, each one being a group of dependent actions.
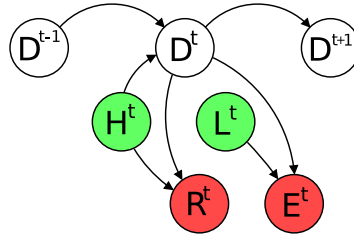
FIGURE 2.5 – Partial representation of the model depicting the relation between the decision D^t, the stimuli H^t and L^t and the actions R^t and E^t.

Example

We define the random variables and their values in the table 2.5 and give a graphical representation in figure 2.6. In order to compare our model to Le Hy's, we define the equivalent model using Le Hy's proposition (see figure 2.7). The independence between some stimuli, decisions and actions reduce the number of parameters of the model. A more detailed analysis is done in the section 2.4.1 - A.

Variable	Definition	Values
High-level stimuli (H)		
H_1	$FoodInMouth$	$(No, Solid, Chewed)$
H_2	$Hunger$	$(Low, Medium, High)$
Low-level stimuli (L)		
L_1	$FoodPosition$	$(Close, Far) \times (Right, Left)$
Reflexive actions (R)		
R_1	$Chew$	(Yes, No)
R_2	$Swallow$	(Yes, No)
External actions (E)		
E_1	$PickFood$	(Yes, No)
E_2	$Walk$	$(Foward, Backward)$
E_3	$Turn$	$(Right, Left)$
Decisions (D)		
D	$Decision$	$(FindFood, Eat)$

TABLE 2.5 – Example of a model following the CHAMELEON proposition.

FIGURE 2.6 – Example of an application of the model. FoodInMouth and Hunger are high-level stimuli, FoodPosition is a low-level stimulus. Chew and Swallow are two reflexive actions. Walk, Turn and PickFood are external actions.
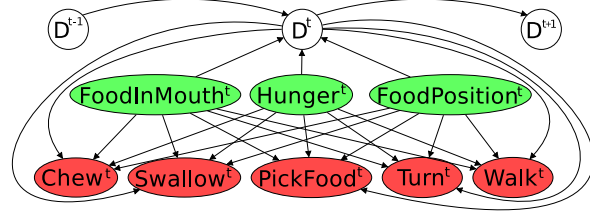
FIGURE 2.7 – Example of a model following Le Hy's specifications and aiming at expressing the same behaviors as the example in figure 2.6.

2.3.1 - B   Attention selection mechanism

Principle

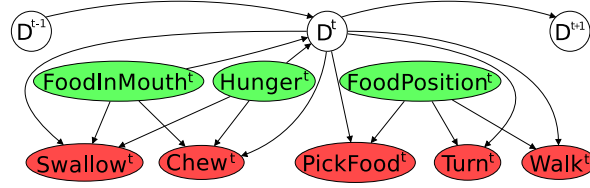The first mechanism used by Le Hy is the *IP* where $P(D^t|D^{t-1}S^t)$ is computed using $P(S_i^t|D^t)$, assuming that all sensors are independent knowing the decision. This hypothesis may be wrong depending on the chosen sensors, and moreover, the more sensors used the higher the chances the hypothesis may be wrong. The second mechanism is the *FEC*. This technique suffers some simple problems: it makes use of probability distributions, but handle them in total opposition with their natural properties. The main aim of this technique is, in the end, to consider a weighted sum of probability distributions, which is easily and rigorously achieved with the sum of random variable over a random index. Therefore, we propose to use instead a mechanism where the agent focus on one high level sensor (H^t) and one low level sensor (L^t). This focus mechanism use two distributions $P(G^t|H^t)$ and $P(J^t|D^tL^t)$ where the random variables G and J gives respectively the index of the high level sensor the agent focuses on and the index of the low level sensor the agent focuses on. As a result, we must simplify the expression of the two distribution because they may take far too much values to be tractable. We propose to express the distributions as follows:

$$\mathbb{P}\left(G^t = i \,\middle|\, H^t\right) = \frac{\theta_i\left(H_i^t\right)}{\sum\limits_n \theta_n\left(H_n^t\right)} \tag{2.1}$$

$$\mathbb{P}\left(J^t = j \,\middle|\, D^t, L^t, K^t\right) = \frac{\lambda\left(D^t, L_j^t\right)}{\sum\limits_m \lambda\left(D^t, L_m^t\right)} \tag{2.2}$$

The higher the values of θ and λ, the more likely the agent will focus on the associated sensor. This reduces greatly the number of parameters still giving the agent a mechanism to focus

only on one sensor. The model uses a very simple algorithm (see figure 2.8).

while agent in world **do**
   h ← agent's high-level stimuli
   l ← agent's low-level stimuli
   Pick i using (2.1)
   Pick d using $\mathbb{P}\left(D^t = d \,\middle|\, D^{t-1} = d^{t-1}, H_i^t = h_i\right)$
   for all $u \in [\![1, N_R]\!]$ **do**
     Pick r using $\mathbb{P}\left(R_u^t = r_u \,\middle|\, D^t = d^t, H_i = h_i\right)$
   end for
   Pick j using (2.2)
   for all $f \in [\![1, N_E]\!]$ **do**
     Pick e using $\mathbb{P}\left(E_f^t = e_f \,\middle|\, D^t = d^t, L_j = l_j\right)$
   end for
   Make avatar do $(r_1, ..., r_{N_R}, e_1, ..., e_{N_E})$
   $d^{t-1} \leftarrow d$
end while

FIGURE 2.8 – Algorithm of the model. It is possible to choose the way the value are picked: randomly following the distribution, using the maximum value in the distribution, etc.

Example

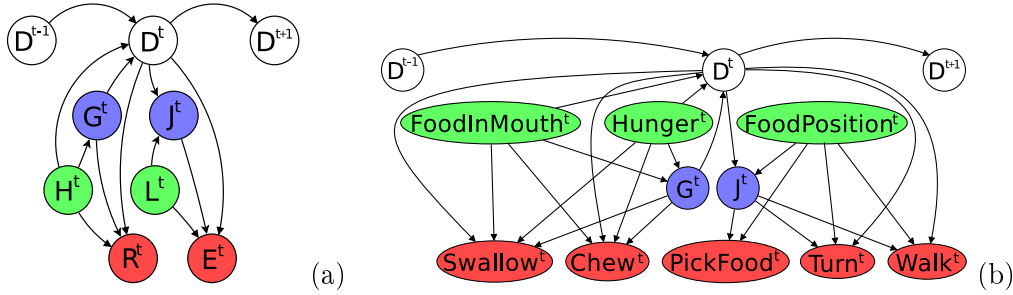A concrete example is given in figure 2.9.

FIGURE 2.9 – In (a), summary of the relation between the random variable of the model. In green the inputs (sensory data), in red the outputs (actions) and in blue the attention variables. In (b), whole model applied to the example defined in section 2.3.1 - A.

2.3.2   Learning the environment: *SGNG*

Models which control virtual humanoids use different types of representation to find paths to go from one point to another. Classic approach use a graph to represent accessible places with nodes and paths between each place by edges, as Le Hy application. Actual solutions tend to use a mesh, with different degrees of complexity, to represent the zones where the humanoid can go. The problem with the latter solution is that it require an algorithm to find the optimal path between two points. The representation must be used directly by Le Hy's model, the graph solution is then more adapted: each node of the graph can be use by the model to attract or push back the humanoid.

To achieve the best believability, we want those nodes to be learned by imitation of a human player instead of being placed *a priori* by a designer. This work as been done in [Thurau et al., 2004b] where nodes and a potential field are learned from humans playing a video game. The agent is then using this representation to move in the game environment, following the field defined at each node. To learn the position of the nodes, Thurau use an algorithm called Growing Neural Gas (GNG).

The *GNG* [Fritzke, 1995] is a graph model which is able of incremental learning. Each node has position in the environment and has a cumulated error which measures how well the node represents its surroundings. Each edge links two nodes and has an age which gives the time it was last activated. This algorithm needs to be omniscient, because the position of the imitated player, the demonstrator, is to be known at any time.

The principle of the *GNG* is to modify its graph, adding or removing nodes and edges and changing the nodes' position for each input of the demonstrator's position. For each input the closest and the second closest nodes are picked. An edge is created between those nodes and the closest node's error is increased. Then the closest nodes and its neighbours are attracted toward the input. All the closest node's edges' age is increased by 1 and too old edges are deleted. Each ζ input a node is inserted between the node with the maximum error and its neighbours having the maximum error. At the end of an iteration, each node's error is decreased by a small amount. The graph stretch and grow to cover the whole space where the player has been monitored to go.

The model has been modified (see figure 2.10) to be able to learn continuously on a player without growing indefinitely but being able to grow if the teacher begins to use a new part of the environment. We called this new version *SGNG* (Stable *GNG*). Instead of inserting a new node each ζ input, a node is inserted when a node's error is superior to a parameter $\overline{Err}$. As each node's error is reduced by a small amount $\grave{E}rr$ for each input, the *SGNG* algorithm does not need a stopping criterion. Indeed, if there are many nodes which represent well the environment, the error added for the input will be small and for a set of inputs, the total added error will be distributed among several nodes. The decreasing of error will avoid new nodes to be added to the *GNG* resulting in a stable state. However if the player which serves as a example, the demonstrator, goes to a place in the environment he has never gone before, the added error will be enough to counter the decay of the error, resulting in new nodes to be created.

The nodes learned by this model can be used directly by the model as low-level stimuli.

Indeed, they represent precise information which will be particularly important for the choice of motion actions. However, the information given by the edges cannot be used as it denotes only proximity between nodes and not a path between them. Two nodes may be linked by an edge because they are close with a obstacle between them.

```
nodes ← {}
edges ← {}
while teacher plays do
   (x,y,z) ← teacher's position
   if |nodes| = 0 or 1 then
      nodes ← nodes ∪ {(x,y,z,error=0)}
   end if
   if |nodes| = 2 then
      edges ← {(nodes,age=0)}
   end if
   n₁ ← closest((x,y,z),nodes)
   n₂ ← secondClosest((x,y,z),nodes)
   edge ← edges ∪ {{n₁,n₂},age=0}}

   n₁.error+=||(x,y,z)-n₁||
   Attract n₁ toward (x,y,z)
   ∀ edge ∈ edgesFrom(n₁), edge.age++
   Delete edges older than Age
   Attract neighbours(n₁) toward (x,y,z)
   ∀ node ∈ nodes, node.error-=Err

   if n₁.error > Err then
      maxErrNei ← maxErrorNeighbour(n₁)
      newNode ← between(n₁,maxErrNei)
      n₁.error/=2
      maxErrNei.error/=2
      newError ← n₁.error+maxErrNei.error
      nodes ← nodes ∪ {(newNode,newError)}
   end if
end while
```

FIGURE 2.10 – Algorithm used to learn the topology of the environment by the $SGNG$

.

2.3.3 Learning the parameters of the model with EM algorithm

In order to learn the parameters of the model, Le Hy uses a modified version of Florez-Larrahondo's IBW algorithm. We prefer not to use this algorithm because it has to approximate very roughly the probabilities computed by the backward procedure. In our case such probabilities give the chances of taking a certain decision at t knowing what the demonstrator did at $t+1...T$. This information should not be lost (for instance, if the demonstrator is

looking for something specific, the learning algorithm cannot know what it is picked up). As a consequence, it seems wiser to learn on a whole sequence of observations (from time 0 to time T) instead of using an incremental version.

EM principle

In our case, the algorithm gathers the values of L_j^t, H_i^t, the stimuli, and R^t, E^t, the actions, at each time step. The values of I^t, J^t and D^t, the hidden states, are not known, thus the data is incomplete. We will apply an *EM* algorithm [Dempster et al., 1977] to be able to learn the model parameters.

The following notation are adopted:

- ▷ T is the length of the sequence of observation used for the learning
- ▷ $\mathcal{A} = \mathcal{A}^{1..T}$, the total sequence of observed actions
- ▷ $\mathcal{S} = \mathcal{S}^{1..T}$, the total sequence of observed stimuli
- ▷ $\mathcal{Q} = \mathcal{Q}^{1..T}$, the total sequence of hidden states
- ▷ Φ is the set of model parameters

Our goal is to find the best model parameters, Φ^*, such as the likelihood, $P(\mathcal{A}|\mathcal{S}, \Phi^*)$, is maximal. This means that we want to find the parameters of the model the most likely to generate an observed sequence given the sensory information: if the model were in place of the observed player, it would most likely generate approximatively the same actions. As we do not have any information about the hidden variables ($\mathcal{Q}$) we choose to use an *EM* to find a local maximum. The idea is to find iteratively a Φ_{n+1} such that $P(\mathcal{A}|\mathcal{S}, \Phi_{n+1}) \geq P(\mathcal{A}|\mathcal{S}, \Phi_n)$.

Finding a sequence of observations

During the introduction of the algorithm we defined the sequences of observation $\mathcal{A}$ and $\mathcal{S}$ of length T. They are the sequences of what the teacher does and perceives. These sequences allow the model to estimate the probability of the hidden states. In our games, avatars "die" disappearing from the environment and "resurrect" reappearing in a random place. Therefore a sequence is the actions and stimuli from the "resurrection" to the "death" of the teacher's avatar. Such sequences usually last from 10 seconds to 5 minutes which is not too long for the algorithm. A last problem concerning the sequences had to be solved: the teacher has a reaction time. Among all the solutions we tried, the most simple worked the best. Instead of associating the actions at t to the stimuli at t, we use the actions at $t + t_{reac}$. The actual value of t_{reac} is discussed in 2.4.3 - A.

Parameters initialization

We choose to stick to the simplest method for the moment: the random initialization. Each parameter is initialized to a random value, then they are normalized for the sum of probability

distributions to be equal to 1. It allows the algorithm to cover many possible solutions at the cost of convergence time.

Stopping criterion

The algorithm needs a stopping criterion, based on the quality of the current set of parameters Φ_n. As the algorithm converges toward a local maximum, we do not know *a priori* the value of the likelihood function at this maximum. We have observed that, as the value of $Q(\Phi_{n+1}|\Phi_n)$ converges, the increase is smaller and smaller at each iteration of the algorithm. We based the stopping criterion on this increase: if $Q(\Phi_{n+1}|\Phi_n) - Q(\Phi_n|\Phi_{n-1}) < w$, the algorithm is stopped and Φ_{n+1} is considered to be the solution.

EM online

Like all the *EM*, our algorithm is offline, which is contrary to our objective. Indeed, the algorithm gives a set of parameters which only satisfies the observed sequence. However, because we have short learning sequences, we can learn from them one after an other, merging the final resulting set of parameters to a global one, Φ_g.

Results

Our learning algorithm allows to learn almost all the parameters by imitation. The attention functions θ_i and λ_j are not yet learned. Our algorithm is much slower than Le Hy's but by avoiding simplistic hypothesis, should give much more accurate results (see section 2.4.3 - B).

2.4   Results

In this section, we present how we adapted our model to the video game *UT2004* and the result for each of the four proposed modifications. The semantic refinement of the model reduces the number of parameters for the model making the learning faster and the model clearer (section 2.4.1 - A). The attention selection mechanism allows the agent to express behaviour which cannot be expressed by Le Hy's model (section 2.4.1 - B). The *SGNG* makes the agent able to adapt rapidly to unknown environments by observing multiple teachers (section 2.4.2). Finally, the imitation learning algorithm allows the agent to evolve rapidly toward a more believable behaviour (section 2.4.3).

2.4.1   Improvements on Le Hy's model

2.4.1 - A   Semantic refinement

The definition of high and low-level stimuli allows the model to reduce the parameters by

$$
\left(\prod_{j=1}^{N_L} L_j\right)|D|^2 + \left(\prod_{i=1}^{N_H}|H_i|\sum_{f=1}^{N_E}|E_f| + \prod_{j=1}^{N_L}|L_j|\sum_{u=1}^{N_R}|R_u|\right)|D| \qquad (2.3)
$$

So the more complex the model, the more favourable is the semantic refinement.

In order to study the consequences of the semantic refinement on a concrete example, we will use our application. The definition of each random variable and the number of values they can take is given in Table 2.6.

Variable	Definition	Number of values
High-level stimuli (H)		
H_1	$Life$	3
H_2	$NumberOfEnemies$	4
H_3	$CurrentWeapon$	14
H_4	$CurrentWeaponAmmunition$	4
H_5	$TakeDamage$	2
H_6	$WeaponsInInventory$	70
Low-level stimuli (L)		
L_1	$Player$	405
L_2	$Weapon$	45
L_3	$Health$	45
L_4	$NavigationPoint$	72
L_5	$RayImpact$	5
Reflexive actions (R)		
R_1	$ChangeWeapon$	12
External actions (E)		
E_1	$Fire$	3
E_2	$Jump$	3
E_3	$Pitch$	3
E_4	Yaw	5
E_5	$Walk$	3
E_6	$Lateral$	3
Decisions (D)		
D	$Decision$	around 10

TABLE 2.6 – Definition of each random variable used in the model applied to the game *UT2004*.

With this example we can now study the number of values needed for the definition of Le Hy's model and CHAMELEON. In order to focus only on the influence of the semantic refine-

ment of the stimuli, we will not consider for now the mechanisms to decrease the complexity of the models: *FEC* and *IP* for Le Hy's model and attention selection for Chameleon.

In our application the gain is worthwhile, the decrease of the number of parameters being between 10 and 92% compared to a model without semantic refinement as show in figure 2.11 and table 2.7. In the game *UT2004*, the number of states, which approximatively corresponds to decisions in our model, is around 10. For 10 decisions the reduction of the number of parameters is around 85%.
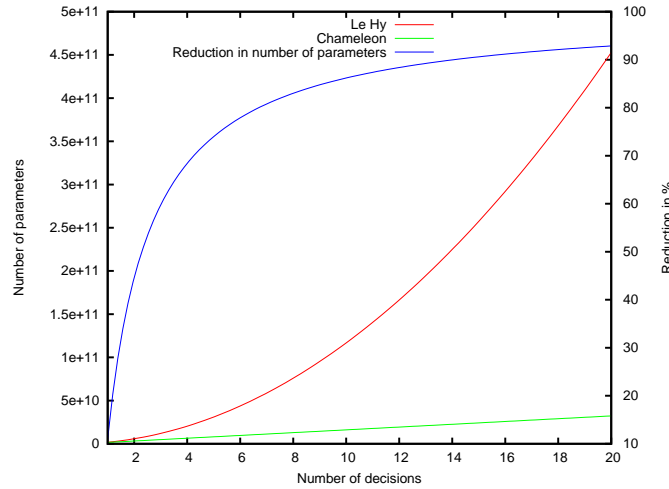
Figure 2.11 – Number of parameters for our application of the model in *UT2004*. The *IP* and *FEC* for Le Hy's model and attention mechanism for Chameleon are not taken into account. The reduction is given compared to Le Hy's model.

	Number of parameters
Full dependence between random variables	3×10^{18}
Independence of actions	8.6×10^{15}
Independence of actions & semantic refinement	6×10^{10}

Table 2.7 – For each hypothesis, the number of values for the definition of the probability distributions with 10 decisions. The semantic refinement allows a noticeable reduction of the number of parameters. The independence of the actions in introduced in Le Hy's work.

The semantic refinement of the model, clearly defining high and low-level stimuli and the associated actions allows an important reduction of the number of values needed for the definition of the probability distributions. This reduction will make the learning faster because less knowledge is to be learn, the independence between variables being already specified. This should allow the agent to adapt even faster answering to the requirement [B10: Fast Evolution].

2.4.1 - B Attention selection mechanism

Number of parameters

The difference in the number of parameters for the previous example is given in Table 2.8 and in figure 2.12. All the modifications we proposed decrease the number of parameters by 36 to 40% compared to Le Hy's model. However, the results for this example cannot be generalised for all the problems. Indeed, the semantic refinement allows an important decrease in the number of parameters whereas the attention selection makes the number of parameters increase. There may be some applications of our model where it needs more parameters than Le Hy's.
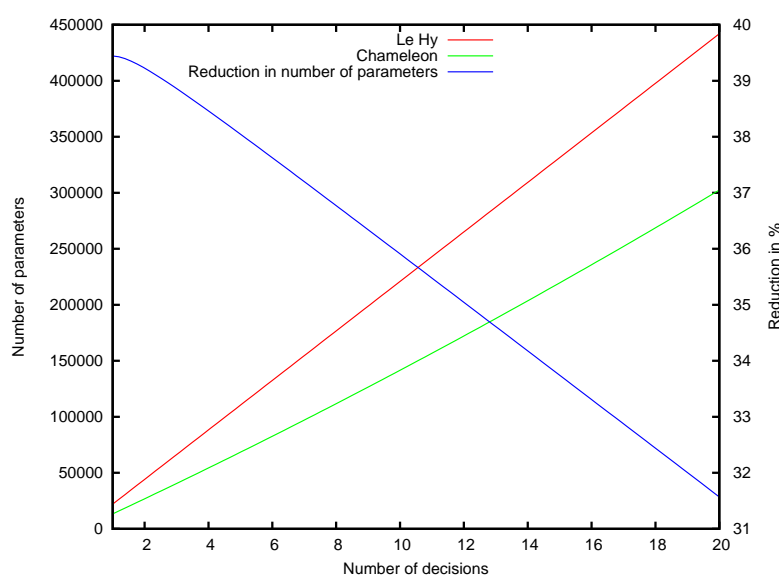
FIGURE 2.12 – Number of parameters for our application of the model in *UT2004*. The reduction is given compared to Le Hy's model.

	No. of parameters
Full dependence between random variables	3×10^{18}
Independence of actions	8.6×10^{15}
Independence of actions & semantic refinement	6×10^{10}
Le Hy: independence of actions & *IP* & *FEC*	2.2×10^5
CHAMELEON: indep. of ac. & sem. ref. & attention	1.4×10^5

TABLE 2.8 – For each hypothesis, the number of values for the definition of the probability distributions with 10 decisions. Our model totals 36% less parameters than Le Hy's model. The *IP* and *FEC* are introduced in Le Hy's thesis and the attention selection mechanism.

Expressiveness

The Le Hy'*FEC* cannot express several simple behaviors. In our example (see figure 2.13), if the attractor are navigation points and the actions forward/backward, the agent will randomly

go back and forth, barely moving because the "expected value" is to stay still. There is also an other problem: if a random distribution is 0 for an action, the product is also 0. For our example, one could set the value for going backward when seeing an attractor in front to 0 and *vice versa*. In this case, the *FEC* cannot be applied because the product would be 0 for all the probabilities.
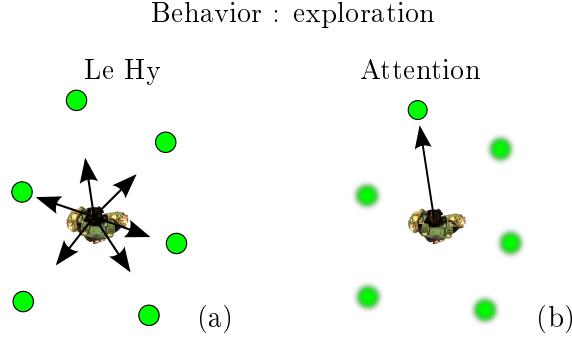
FIGURE 2.13 – In (a) an illustration of the problem with *FEC*: it does not give a believable action because the agent may constantly switch between the two attractors, oscillating constantly. In (b) the attention selection mechanism produces a better distribution in term of believability: the attention gives a believable action because the agent focus on the attractor ahead instead of switching between the two attractors like the *FEC* would do.

2.4.2 Learning the environment: *SGNG*

Results 2D/3D

We trained 2 *SGNG* on 2 different maps. The first one is a simple map, called Training Day, it is small and flat which is interesting to visualize the data in 2 dimensions. The second one, called Mixer, is much bigger and complex with stairs, elevators and slopes which is interesting to see if the *SGNG* behave well in 3 dimensions. The results is given in figure 2.14.
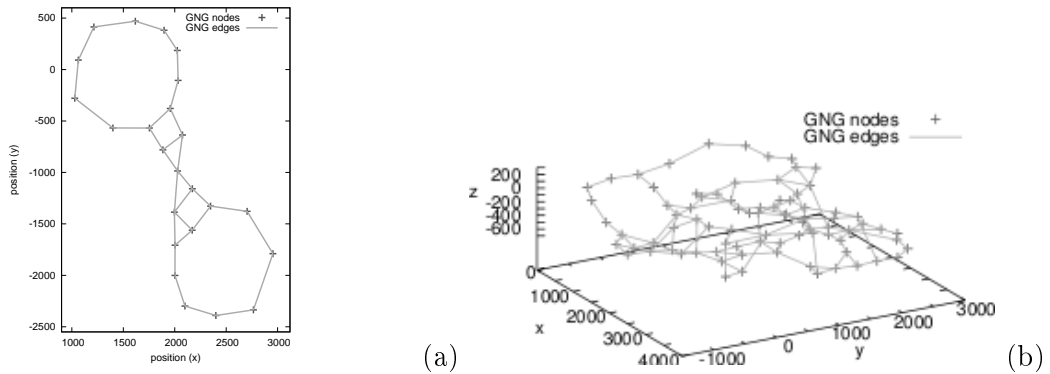
FIGURE 2.14 – Result of a growing neural gas learned from a player for a simple map, top view (a) and from a player for a complex map (b).

Similarity with manually

To study the quality of the learned topology, we first chose to compare the *SGNG*'s nodes with the navigation point placed manually by the map creators (see figure 2.15). Of course, we do not want the *SGNG* to fit exactly those points but it gives a first evaluation of the learned representation. The two representations look alike which indicates that the model is very effective in learning the shape of the map. However, there are zones where the *SGNG*'s nodes are more concentrated than the navigation points and other where they are less concentrated. We cannot tell now if it is a good behavior or not as we should evaluate an agent using this representation to see if it navigate well. Even in the less concentrated zones, the nodes are always close enough to be seen from one to another, so it should not be a problem.
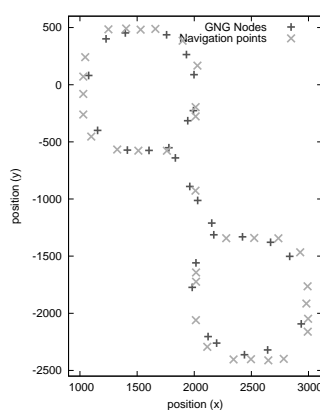
FIGURE 2.15 – Comparison of nodes learned by the *SGNG* with the navigation points placed manually by the game developers.

Time evolution

To study the time evolution of the *SGNG*'s characteristics, we introduce a distance measure: the sum of the distance between each navigation point and its closest node. We also study the evolution of the number of nodes because we do not want the *SGNG* to grow indefinitely. Figure 2.16(a) shows this two measures for the simple and the complex maps. For the simple map, the *SGNG* reached its maximum number of node and minimum error in approximatively 5 minutes of real-time simulation. For the complex map, it takes more time, about 25 minutes, but results at 12 minutes are quite good. Those results show that it is possible to have an agent learn during the play.

Learning with multiple professors

The *SGNG* can handle inputs from multiple professor. Figure 2.16(b) shows the distance and number of node for a *SGNG* trained on 1 professor and for a *SGNG* trained on 4 professors. The learning with 4 professors is, as expected, faster: about 3 minutes for the distance to stabilize instead of 5 minutes for 1 professor. It is interesting to note that the learning is not

FIGURE 2.16 – Time evolution of the cumulated distance to navigation points defined manually and the *SGNG*' nodes and the *SGNG*' number of nodes.

4 times faster but the gain is still important. Learning with multiple professors seems to give a *SGNG* with less variation during the learning. The gain have however a small drawback: the number of nodes is a little superior for multiple professors. It may be due to the fact that professors are scattered in the environment instead of a unique professor following a path.
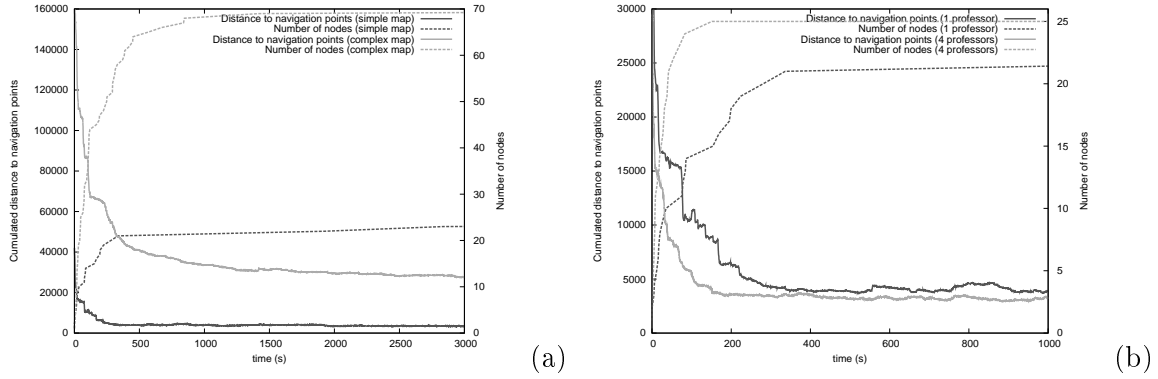
Comparison GNG/SGNG

Figure 2.17 a comparison betwenn the *GNG* and the *SGNG*, using three measures during the learning for the simple environment. Sensitivity measures how much the *SGNG* successfully represents the part of the environment the teacher used which can be seen as true positives. The higher the value the better the *SGNG* is. Specificity measures how much the *SGNG* did not represent the part of the environment the teacher did not use which can be seen as true negatives. The higher the value, the better the *SGNG* is. We also study the number of nodes the *SGNG* has because we do not want the *SGNG* to have either too many or too few nodes.

2.4.3   Learning the parameters of the model with EM algorithm

In this section, we first study the influence of the data given to the *EM*: the observation sequences and the model parameters (section 2.4.3 - A). Then we study the convergence of a learning on one sequence and multiple sequences (section 2.4.3 - B). Finally we try to evaluate the resulting behaviors with different objective and subjective measures (section 2.4.3 - C).

2.4.3 - A   Impact of the EM and Parameters on the Results

Impact of the teacher's reaction time

When the teacher is observed by the learning algorithm, a snapshot of the values of the stimuli and the actions is taken at each time step t. However, the teacher actions at time t may not
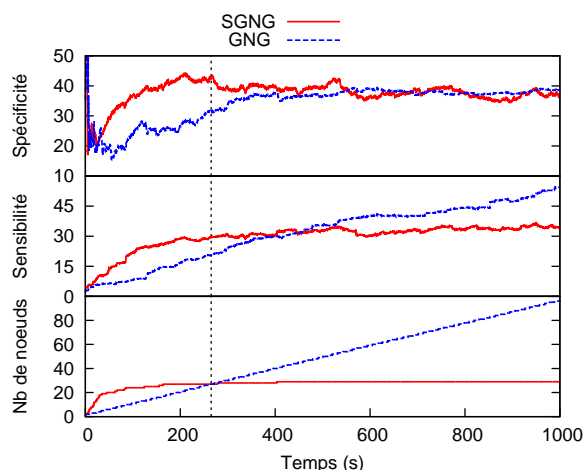
FIGURE 2.17 – Comparison *GNG/SGNG*

reflect a reaction to the stimuli at time t. We must then take into account the reaction time of the teacher, allowing our model to find the relation between stimuli and actions which are actually related.

The reaction time may not be the same between individuals and may also vary for one individual over the time. However, we find that the simplest solution, using a constant reaction time for all the teachers, give the best results. Other methods were tried, like associating the more likely action to the stimuli according to the current model parameters, or aligning variations in both stimuli and actions but they did not make the learning algorithm converges toward parameters more likely to generate the observations. The choice of the reaction time for a game has to be done once and for all using a graph like figure 2.18.
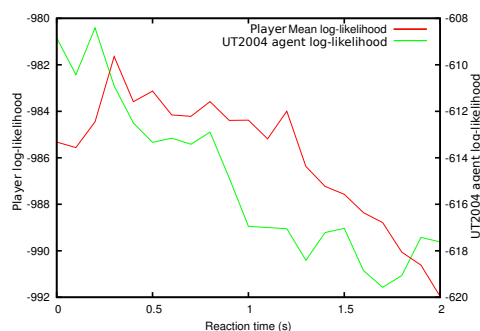
FIGURE 2.18 – Mean log-likelihood of the final result after learning on 105 different sequences of observations of the behavior of a player and 100 different sequences of observations for a *UT2004* agent. The reaction time varies from 0 second to 2 seconds and the model has 10 decisions.

According to figure 2.18, the reaction time of a player is very variable. The reaction time of 300ms gives the maximum likelihood, but the difference is not very important with the likelihood found for reaction times between 400ms and 800ms. For now, we will use a reaction time of 300ms in the following experiments, but a variable time of reaction could improve the

results.

In order to show that the results of the learning are really impacted by the reaction time, we studied the variations of the log-likelihood depending on the reaction time for a *UT2004* agent teacher (all the other experiments are done with human teachers). Figure 2.18 show that best log-likelihood are achieved for reaction times between 0 and 200 ms. These results are coherent with the fact that *UT2004* agents do not model the reaction time. It also shows that the players' behaviors are much more complicated to learn, their reaction time being much more variable.

Impact of the number of decisions

The mechanism of decisions allows the agent to produce logical sequences of actions, to make actions according to its needs and to simulate a short time memory. As expected, the figure 2.19 shows that the more the decisions, the better the model can express the observed behaviors. Indeed, the model is more complex and can make the agent behave in a more subtle way. However, as the number of parameters increase with the number of decisions, the time needed for the algorithm to converge is longer. Too many decisions should be avoided to make the learning fast and fulfil the requirement [B10: Fast Evolution]. According to the results, increasing the number of decisions over 10 does not improve the results in a significant way. We remind the reader that agents originally coded in the *UT2004* game use approximatively 10 main states.
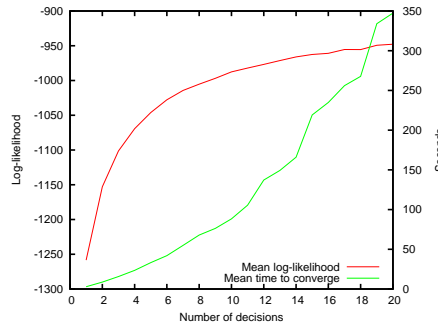
FIGURE 2.19 – Mean log-likelihood of the final result and time to converge for 105 different sequences of observations. The number of decisions varies from 1 to 20 with a fixed reaction time of 300ms.

2.4.3 - B   Characteristics of the EM

According to the two previous studies, we will use 10 decisions and a reaction time of 300ms for the following experiments. The function Q, is the function which is optimized by the *EM* algorithm. The higher it is, the higher is the log-likelihood. Figure 2.20 shows the value of the total log-likelihood for each iteration of the *EM*. The first iterations make the value increase sharply, after 10 iterations the increase becomes very slow, stabilising between -500 and -300. As the value is a log, this likelihood is very small. We can also see that some learnings finish faster, in about 30 iterations, while others take around 100 iterations.
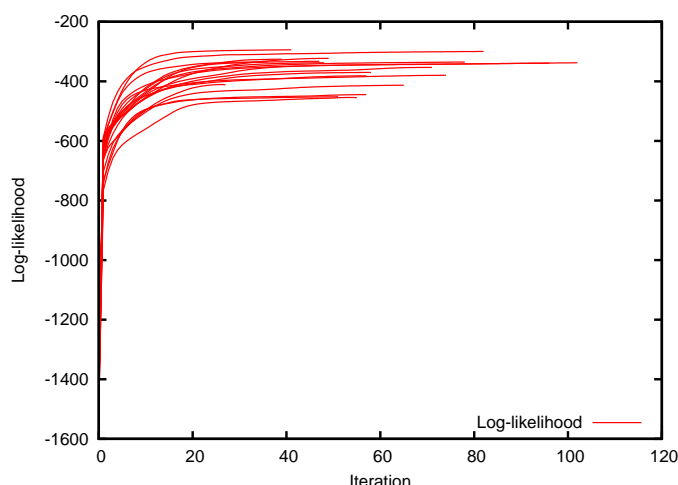
FIGURE 2.20 – Time evolution of the total log-likelihood for 20 learnings on sequences of same length. Each learning starts the same initial distributions.

2.4.3 - C   Resulting behaviors

The final goal of the four propositions (semantic refinement, attention selection, learning of the environment and learning of the behavior) is to make the agent produce believable behaviors. In the following experiments, the model learned on one unique teacher using 10 decisions. It also considered that the teacher had a constant reaction time of 300ms. The results given are after learning on the teacher during 40 minutes. The teacher played against a *UT2004* agent in the environment named in the game *Training Day*.

Study of the distributions

Before analysing the whole behavior we can study the distributions of actions to see if they really look like the ones in [Le Hy, 2007, page 47, in French].

For the same decision, the distributions shown in figure 2.21 confirms that the agent does not react in the same way for different positions of a same object. In the example, if a weapon is at the left, close and at the same height as the agent (left figure), the agent will go forward, move laterally left and not turn. If the weapon is at the right and at the same height (right figure), the agent will probably not turn go forward and move laterally right. In the two cases, the agent will look in the direction of the horizon which is the direction of the weapon in term of pitch. In the two cases, the agent will surely pick the weapon.

For the same stimulus but different decisions, the agent may act differently. The figure 2.22 shows the distributions for an enemy player on the right of the agent, moving to the right, at the same height and at an average distance for two different decisions. In the first decision (left figure), the agent moves forward and turn right in order to aim at the enemy and reduce the distance to the player. In the second decision, the agent turns also right but may move forward or backward and move laterally to the left in order to aim at the enemy but keep the same distance to the player. In the two cases, the agent looks at the direction
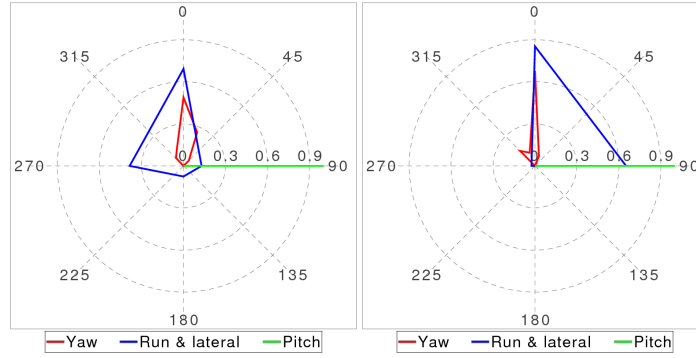
FIGURE 2.21 – Two distributions of movement actions for the same decision and same kind of stimuli but different values. The left graph is for a stimulus representing a weapon on the left of the agent and very close. The right graph is for a stimulus representing also a weapon but it is on the right of the agent and is also very close. The weapon is about the same height as the agent.

of the horizon as it is the direction of the enemy.
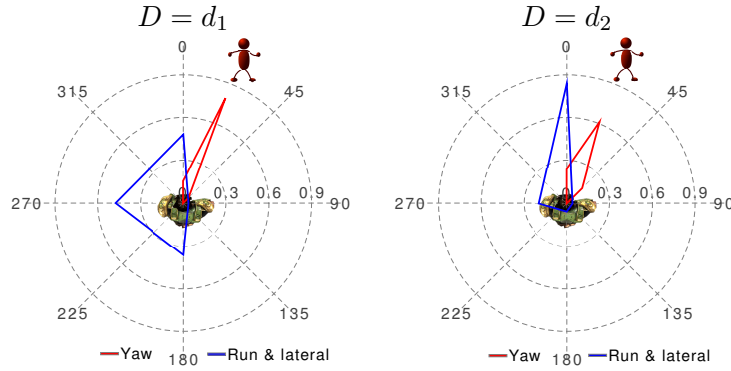
FIGURE 2.22 – Two distributions of movement actions for the same stimulus but different decisions. The two graphs are for stimulus representing an enemy player, right of the agent, moving to the right of the agent and at an average distance.

The study of the movement distributions shows that some knowledge is assimilated by the model parameters. The role of the decision becomes obvious with different tactics being used according to the state of the agent. However some distributions do not represent a believable behavior at all. The reasons can be many: problem with reaction time, bad attention values, etc.

Signatures

In a previous study [Tence and Buche, 2008], we presented a method to spot differences in the behavior of players and agents. While it is not proved to be an indicator of believability, it may be used to spot non-believability. The idea is to monitor the movements of avatars and to extract some statistics. These statistics are the signatures of the behaviors, similarities can be spotted between agents and between players. The most important information is the

differences spotted between players and agents which often reveal problem in the behavior of the agent. It is also possible to represent the distance between the signatures, visualizing the similarities in a more natural way. This distance allows the measure to take into account that turning 80° right and 90° right is almost the same but 90° right and 90° left is very different. So as to visualise the distance we represent the signatures on a plane using the *MDS* method. Both the *EMD* and *MDS* are detailed in [Tence and Buche, 2008]. The idea is that close (Euclidean distance) representations of the signatures in the *MDS* are close in in the *EMD* distance and thus are similar, the contrary being also true. As it is only a question of relative distance to the other, the graphs do not have any tics or values on the axis. In order to have a more complete study of the signatures we represent the signatures for the CHAMELEON and the Le Hy agent as well as seven different players and nine *UT2004* agents, each with a different skill level.
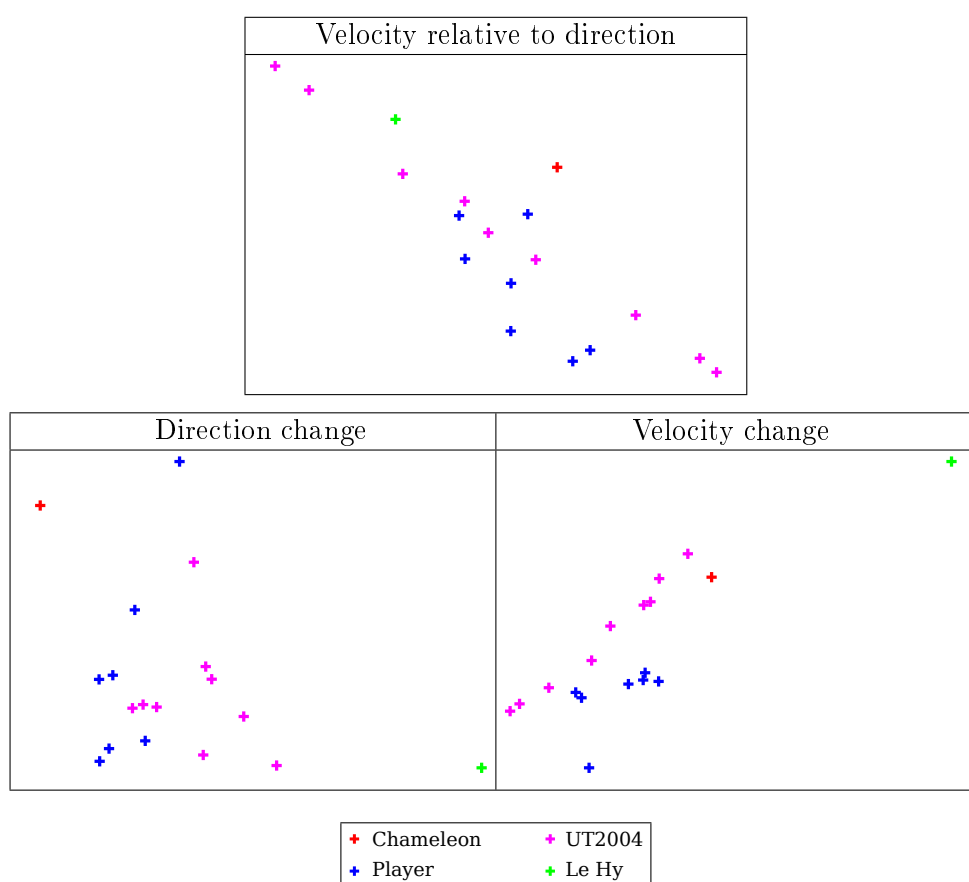
FIGURE 2.23 – *EMD* between the signatures represented using the *MDS* for one CHAMELEON, one Le Hy agent, seven different players and nine *UT2004* agent, each one with a different skill level. The correlation factors for the *MDS* for all the representations are very high (> 0.98).

For the first graph (figure 2.23, top), the *UT2004* agent are widespread. The medium skilled ones are close to the players. Le Hy agent is pretty far from the players and CHAMELEON is not very far but not in the "cloud" of the players. That means it may be taken for a player but not an average one. In the second graph (figure 2.23, bottom left), the players are quite widespread and the *UT2004* agents are still quite close. Le Hy agent is very far from the

players and CHAMELEON is a bit far also. Finally, for the last graph (figure 2.23, bottom right), Le Hy agent is very far from the players, CHAMELEON is quite far and the agents are not very far but can be clearly separated from the players. To conclude, it appears that some *UT2004* agents are the closest to the players, then comes CHAMELEON and then Le Hy's agent. The fact the agent from the game are this close is because character designers used a lot of time improving the way their agent moves. However, it may not apply to the behaviors.

It seems that CHAMELEON managed to perform better than Le Hy's agent. Because the discretization of the actions and the interface between the game and the model can be greatly improved, *UT2004* agents perform better than CHAMELEON. As all the details of the agent's behavior are important, this should be improved on CHAMELEON in order to be able to fool the players into thinking CHAMELEON is an other player.

2.5   Conclusion

This chapter aims at designing a Behavior model for the control of believable characters in video games. The character is controlled by a computer program we call an agent. We define a believable agent as a computer program able to control a virtual body in a virtual environment so that other human users in the environment think the virtual body is controlled by another human user. As this definition is pretty vague, we define 10 requirements for a character to be believable, based on previous experiments and work: reaction, reaction time, variability, unpredictability, understandability, human-like perceptions, planning abilities, memory evolution and fast evolution.

In order to fulfil these requirements, we studied the existing Behavior models developed both in the research and the industry. We grouped them into four types: connectionist models, state transition systems, production systems and probabilistic models, each one having its strengths and weaknesses. As one of the requirement is that the model is able to evolve, we had to find learning algorithms for the Behavior model. We find out that imitation is the best way to believability. With these studies in mind we find out that the Behavior model developed by Le Hy in his thesis [Le Hy, 2007] answers to most of the requirements but has still some limitations.

In this chapter we propose four modifications or replacements to parts of Le Hy's model. We first try to reduce the number of parameters in the model with a semantic refinement. Then we replace the two mechanisms to break the complexity of the probability distributions by an attention selection mechanism. This avoid the agent switching constantly between stimuli. We add to the model the ability to learn by imitation the layout of environments with a model named Growing Neural Network. Finally we totally revamp the learning algorithm with an Expectation-Maximization method.

The proposition makes the model able to learn how to act in the environment rapidly. Stimulus-action associations are made which the agent look-like a human player. However the learning also learn wrong associations which destroy the illusion of believability. According to our studies, our model perform better than Le Hy's but work has still to be done on the model to achieve the final goal.

The next step is to evaluate our work. As believability is subjective, evaluation is a critical and complex step. Even if it was not intended to, Turing's test is still considered as a reference for believability evaluation [Turing, 1950a]. In its standard interpretation, a judge must chat with a human and a machine using only text. If, after a certain amount of time, the judge cannot tell which one is artificial, the machine is said to be intelligent. This test's goal was to assess intelligence but it has been much criticized [Searle, 1980,Hayes and Ford, 1995]. This critique, however, does not apply to believability and it is even a very good basis for assessing believability as we defined earlier.

There are many parameters for believability evaluation methods [Mac Namee, 2004] [Gorman et al., 2006b,Livingstone, 2006]. The first one is to cast or not to cast doubt on the nature of the presented character(s). This choice is often linked with mixing agents and humans so that the judges assess real humans' believability too. This can be useful to avoid bias induced by prejudices and to have a reference: humans usually do not score a perfect believability. Another parameter is the number of questions and answers. Turing's test features only one question and a yes/no answer whereas other tests feature many questions and scales to answer. The former choice may be too restrictive while the latter may result in too much undecided answer to "beat" the test. Another problem is the computation of the overall believability score which, in case of multiple questions, may give experimenters too much influence on the results. To add more objectivity, it is possible to have relative scoring instead of absolute: the score given to a example can answer to "is example A better than example B?". It is also necessary to decide if judges are players or only spectators. While players can actively test evaluated characters, spectators are more focused on them and can notice much more details in the behaviors. Finally the choice of the judges is really important. Cultural origins [Mac Namee, 2004] and level of experience [Bossard et al., 2009] may have a noticeable impact on believability scores.

Contents of Chapter 3

Chapter 3

JABU — anticipatory behavior in virtual universe

This chapter deals with believable simulations of behavior. To be believable, virtual entities must be equipped with the ability to anticipate, i.e. to predict the behavior of the other entities and the subsequent consequences on the environment. For that purpose, we propose an original approach where the entity possesses an autonomous world of simulation within simulation, in which it can simulate itself (with its own model of behavior) and simulate the environment (with the representation of the behaviors of the other entities). This principle is illustrated by the development of an artificial juggler in 3D. In this application, the juggler predicts the motion of the balls in the air and uses its predictions to coordinate its own behavior in order to continue to juggle. Furthermore, an interface allows the human user to launch an additional ball that the juggler must intercept and add to the balls it is already juggling.

3.1   Introduction

AUTOMATIC decision-making by artificial systems situated within given environments is difficult to model, especially when the environment is dynamic, complex, open, and populated with independent entities. Virtual reality is an ideal field for modeling complex decision-making behavior, as it occurs at the heart of interactions with humans, who elicit subtle and varied reactions and perceptions. The emergence of interactive motion systems, (Wiimote, Kinect, etc.) makes studying the subtlety of these connections even more crucial. Graphical realism is not sufficient, and indeed is no longer the priority: the reactions and thus real-time decision-making by the virtual entities within the environments must be "believable" [Bates, 1992a]. This notion of believability is subtle and varied and can be studied according to many different criteria. For example, there are studies relating to movement [Van Welbergen et al., 2009], to the character's realism [Ho et al., 2008], and to the impact of the synchronization between image/sound/movement [Groom et al., 2009b]. We, however, are interested in the believability of a behavior "during a behavioral interaction" and more exactly, in the ability to anticipate the futur of the environment thanks to the knowledge

of its dynamic properties. We don't adress graphical realism neither gesture synthesis. Our work focuses on the perception-based decision-making dynamic.

Anticipation can be observed in living creatures at almost every stage of evolution (including bacteria, cells, plants, vertebrates and mammals). To illustrate this principle, the "waiter experiment" [Hugon et al., 1982, Dufossé et al., 1985] gives the example of a waiter holding a tray in one hand, on which is balanced a jug of water. With his other hand, the waiter lifts the jug. At that moment, the hand holding the tray, which should have lifted as it is now carrying less weight, remains in the same position. Instinctively, the waiter predicts the consequences of his movement, **anticipating** the relief of the load-bearing arm [Berthoz, 1997].

Surprisingly, despite increasing (or even omnipresent) proof of its importance, anticipation has for a long time been ignored or underestimated in the behavioral modeling of virtual entities. However, studies, particularly in psychology [Hoffmann et al., 2007, Pezzulo et al., 2007b] and neurology [Rizzolatti et al., 1996, Hesslow, 2002], stress the importance of our anticipation mechanisms in reasoning. They also point to the use of internal behavioral simulations in the lead-up to effective reasoning [Berthoz, 1997]. As this behavior-modeling paradigm is relatively new, it remains unconfirmed as there are still many questions regarding its use, functioning and success. This document addresses these questions by proposing to model anticipation as an internal simulation of the evolution of the environment and of the interactions between the autonomous entity and that environment. Internal simulation such as this is thus taken into account during the decision-making process.

This chapter begins by presenting the theoretical framework of anticipation in decision-making (section 3.2). We then go on to examine the use of internal behavioral simulation in integrating the anticipatory process into an artificial entity's decision-making process (section 3.3). Next, we describe our application, in which a virtual juggler anticipates the trajectory of balls in a simulated model, and subsequently controls the position of his hands (section 3.4). We then evaluate the impact of anticipation on the virtual juggler's decision-making processes (section 3.5). Finally, we present our conclusions and the future direction our work will take (section 3.6).

3.2 Theoretical framework: the role of anticipation in decision-making

3.2.1 The foundations of anticipation

As our aim is to simulate human-like behaviors, it is important to have a look at previous research in different fields of cognitive science like philosophy, psychology, neurology and physiology.

3.2.1 - A Philosophy: man is oriented toward his future

The first people to study human behavior were philosophers. These studies showed that Man, by his very nature, is oriented toward the future, questioning his future. Philosophers suggest two approaches: (i) the use of past knowledge in anticipating the future [Epicure, 200, Bergson, 1896], an idea which is also prevalent in psychology (section 3.2.1 - B), and (ii) the theory of simulation [Hume, 1740, Husserl, 1992], which is also prevalent in neurology and physiology (section 3.2.1 - C).

3.2.1 - B Psychology: the role of past knowledge in anticipating the future

This idea suggests that we use the memories of our past experiences and observations in order to anticipate the consequences of our actions and the behavior of those around us. Identifying regularly observed phenomena from the past enables us to react in an anticipatory manner. This idea opened the door to anticipation studies in behavioral psychology. Studies on rats led to the identification of anticipatory mechanisms [Colwill and Rescorla, 1985] and the use of predictive models[1] of the environment [Tolman, 1959]. Studies with human subjects revealed the importance of anticipation in human behavior [Kunde et al., 2002, Kunde et al., 2004]. Hoffman thus put forward a behavioral model (Anticipatory Behavioral Control: ABC), basing his work on the ideomotor principle [2] [Hoffmann, 2003]. ABC is a theoretical decision-making and learning model in which the subject first focuses on the desired outcome, and then takes the context into account in order to choose a suitable action.

3.2.1 - C Neurology and physiology: simulation theory

For this field, the brain is a simulator for actions, and thought as simulated interaction with the environment [Hesslow, 2002, Berthoz, 1997, Berthoz, 2003]. It was also a precursor to all anticipation studies. Cerebral imaging techniques in neurology have enabled to measure cerebral activity at the heart of a monkey or a human's brain whilst carrying out certain actions. It was possible to use the results of such tests to isolate an area of the brain known as mirror neurons [Rizzolatti et al., 1996]. These neurons are activated in very similar ways in the situations described in table 3.1. Although all these elements are still hotly debated topics in cognitive science, they led some researchers to suggest that this area of the brain might enable the mental simulation of actions and the anticipation of others' behavior via an empathy mechanism.

Internal simulation, particularly of movement, has foundations in neurophysiology which are now well documented [Brunia, 1999]. Individuals do not make these sensorimotor predictions through logical reasoning based on abstract symbols representing the real world. Instead, they are made via biological simulation where, thanks to inhibiting mechanisms, "everything occurs as if" the individuals were really acting [Berthoz, 1997]. For vision, for example, the

---

[1] Predictive models suggest potential outcomes, thus defining the anticipation process

[2] The ideomotor principle suggests that actions are chosen according to the desired outcomes rather than as a reaction to a stimuli

Carrying out an action	
Imagining carrying out the same action	
Watching a third person carrying out the same action	
Imagining or anticipating a third person carrying out the same action	

TABLE 3.1 – Situations "showing" the mental simulation of actions and the anticipation of others' behavior in the brain.

brain has a way of imagining eye movements without moving thanks to the action of inhibiting neurons which close the command circuit of the ocular muscles: by staring at a spot in front of you, and by moving one's attention, one has the impression of looking around the room, a sort of "interior regard". This virtual eye moment is simulated by the brain activating the same neurons, except that the action of the motor neurons has been inhibited. The brain can thus be considered a *biological simulator* which is able to make predictions based on memory and by hypothesizing based on internal models of the phenomenon.

3.2.2 The importance of an explicit anticipation model

As we have already seen, anticipation is considered a key factor in defining decision-making and indeed for behavior in general. We must now wonder how an explicit modelling of such a mechanism might benefit our research, so that we might be aware of believable behavior in interactive simulations. To better understand our position, we need to clearly differentiate implicit and explicit anticipation model.

3.2.2 - A   Implicit anticipation model

In the first case, implicit anticipation does not rely on specific predictive models in order to anticipate the future; obtaining knowledge about the future is part of the decision-making mechanism, or of genetic information. One example of low-level implicit anticipations is that of trees which shed their leaves in autumn in order to avoid frost damage in winter. As temperatures drop and days get shorter in autumn, the trees anticipate the arrival of winter and duly react by shedding their leaves, breaking the connection at the inside of the leaf stems (so that the leaves can then be carried away by the wind). It is likely that the tree does not use an explicit environmental model to predict the coming of winter but rather genetically-transmitted implicit anticipation. The same anticipatory mechanisms can be observed in

hibernating animals.

3.2.2 - B   Explicit anticipation model

In the second case, explicit anticipation uses one or more explicit predictive model of the environment and/or self and uses these models to make predictions about the future. An example compares the hunting behavior of a dog with that of a snake [Riegler, 2001]. It has been observed that dogs go on chasing prey even if they can no longer see it, by using a predictive prey model thus predicting its behavior and continuing with the hunt. However, when snakes lose sight of their prey, they cannot predict their movements or future positions, and instead anticipate (implicitly), that they have more chance of catching their prey by returning to the place where they last saw it. According to Rosen, human behavior is essentially anticipatory and is based on explicit environmental models [Rosen, 1985a]. He offers the example of a hunter who finds himself a few meters away from a bear, and whose behavior would be to hide so as not to be seen. It is not the sight of the bear itself which triggers this reaction but rather that which the hunter imagines, or anticipates, might happen as a result of an encounter with a bear.

3.2.2 - C   Conclusions

From these examples, it would seem that complex cognitive behaviors (of humans and intelligent animals) rely on explicit predictive models used to anticipate their environment, whereas less complex cognitive behaviors do not. As our aim is to simulate artificial human-like behaviors, we shall use explicit anticipatory models to develop our artificial behaviors. Indeed, an implicit model would risk not being able to account for the complexity and the subtleties of human decision-making.

3.2.3   Anticipation-based artificial behavior

In this section, we will examine the integration of anticipatory mechanisms with explicit representations in artificial systems. This anticipatory approach was long ignored, or at least underestimated, in the field of computing and artificial systems. Despite the large quantity of research relating to anticipation (planning, goal-oriented behavior, reinforcement learning, latent learning, etc.), studies focusing on anticipation are relatively new.

3.2.3 - A   Anticipation in studies of artificial systems

Rosen was one of the first to focus on anticipation in studying modeled artificial behaviors. In his founding work he introduced the first elements, issues and definitions of this approach [Rosen, 1985a]. Following this work, the idea took many years to take off in the scientific community, who worked first in isolation and then, from 2002, within a number of workgroups. The annual CASYS (International journal of computing anticipatory systems)

conference, the biannual ABIALS workshop (Anticipatory Behavior In Adaptive Learning Systems), the European Project Mind RACES (from Reactive to Anticipatory Cognitive Embodied Systems), and the creation of the institute for research in anticipatory systems (university of Texas in Dallas) to name but a few.

3.2.3 - B Definitions

Thanks to the sudden interest in anticipatory systems we are now able to establish some semantic details. The first definition of an anticipatory system was proposed by Rosen: *"A system containing a predictive model of itself and/or its environment, enabling it to change state at any given instant in accordance with the predictions made by the model concerning a given moment in the future"* [Rosen, 1985a]. The anticipatory system will therefore use the predictive model or models in order to obtain information about the potential future and thus be able to make a decision using present, past, and future information.

Another more general definition is offered by Butz [Butz et al., 2007]. An anticipatory behavior is *"a process or behavior which does not only depend on the past and present but also includes predictions, expectations and beliefs concerning the future"*. This definition is more general in the sense that it does not necessarily include the use of predictive models but it implies the use of future information in decision-making in the present.

Integrating an anticipatory mechanism will therefore help in *improving decision-making* in our virtual entities. Indeed, such predictions could be compared to real effects. The entity could thus refine its predictions through *learning*. These two criteria will enable the entities to adapt more quickly or effectively to their environments [Butz et al., 2007], therefore improving their behavioral believability.

3.2.3 - C Classification: anticipating sensory and state considerations

As we have already seen, the definitions of anticipatory systems are very general and simply express the fact that these systems try to obtain future information about their environment in order to use them to improve their decision-making in the present. We must therefore try to clarify the subject using certain classifications through the explicit study of anticipation.

There are three categories of explicit anticipation [Butz et al., 2003b]:

1. the anticipation of considerations which brings together approaches using predictions about the possible *rewards of the potential actions*. In this category we find all the reinforcement learning algorithms, Q-learning [Watkins, 1989], Sarsa [Rummery and Niranjan, 1994], classifiers systems [Sigaud and Wilson, 2007b].

2. sensory anticipation includes the use of predictive environmental models to *orient the entities' perceptions* more effectively, especially in order to process expected rather than sudden perceptions [Brunia, 1999]. This approach brings together the ideas of active perception, attention and sensory blindness. A good example of this is the experiment by Simon and Chabris, which asked students to watch a basketball match on television

and to count the number of passes between players [Simons and Chabris, 1999]. Almost all of the students gave the correct answer. However, the majority failed to notice the man dressed as a gorilla who walked into view, stopping to beat his fists against his chest.

3. state anticipation deals with the use of predictive models to *foresee evolutions in the environment* in order to observe how this is taken into account in decision-making. Virtual entities could use this knowledge in order to act in goal-oriented anticipation (planning), *i.e.* to try to detect these unwanted states in the environment before they occur and thus react so that they might be avoided [Davidsson, 2003a].

It must be noted that state anticipation can also include anticipation of considerations and sensory anticipation and as a consequence seems especially interesting. It has been the focus of a great deal of research [Meyer, 1999, Laird, 2001c, Davidsson, 2003a, Labbé and Sigaud, 2004b, Johansson and Balkenius, 2007, Buche et al., 2010b]. These studies raise the following questions:

▷ In what circumstances are anticipatory behaviors the most believable and lead to faster adaptation than non-anticipatory processes?

▷ What is the link between immediate decision-making and the anticipatory mechanism?

▷ What are the links between anticipation and learning?

The goal of this chapter is to give clues to these questions. First, we propose an anticipatory architecture model (section 3.3). Next, we show its application to a virtual juggler (section 3.4). Finally, we evaluate the anticipatory mechanism, its qualities, and its impact on decision-making for the juggler animation (section 3.5). It must be noted that the virtual juggler is just an application to test our predictive model.

3.3 Proposal : conceptual framework

Our proposal is based on the theory of internal simulation with explicit anticipatory representation (see section 3.2.1 - C), *i.e.* advance simulation of the evolution of the entity's environment (state representation, see section 3.2.3 - C) in order to make a decision. To do so, we propose to populate a virtual world with our virtual entities with the ability to predict (section 3.3.1) and the ability to learn (section 3.3.2). The aim is to create an explicit anticipatory model, the most important issue is to achieve a final behavior which accounts for the characteristics of believability and adaptation.

3.3.1 Virtual actors and imaginary worlds

Whilst acting within the virtual world, each entity can simulate its own behavior in its imaginary world (with its own behavioral model), along with that of its environment (using the

representation that it has of the behavior of other entities). This simulation occurs a phase ahead of the original simulation, enabling the entities to make predictions. This imaginary space, unique to each entity, functions in parallel with its activity within the virtual world, asynchronously so as not to block the behavioral animation (see figure 3.1). This imaginary world is a universe of *simulation within the simulation.*
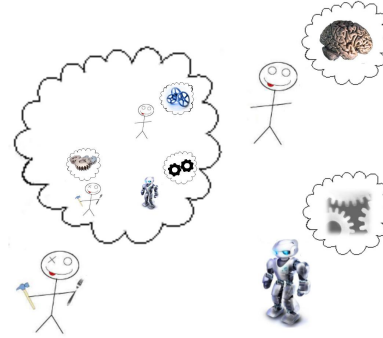
FIGURE 3.1 – The entity (left) anticipates the behavior of the other entities (right). In order to do so, it possesses an imaginary world in which it "imagines" what it going to happen.

3.3.2 Virtual actors and learning

Prediction in the imaginary word implies a representation of this world and of its dynamic. To obtain this representation can be a hard challenge because a world adapted to this approach is open: unpredicable interactions can appear every time and moreover the dynamic properties can be disturbed (for instance, wind can disrupte the trajectories of flying objects). So, learning mechanisms are a good way to learn the dynamic of the world.

In our proposal, predictions in the imaginary world are improved by observing the virtual world online. The virtual actor will then modify its representations of other entities using a learning mechanism. It must be noted that this observed world can also be populated with other actors, or with human-controlled avatars [Stoffregen et al., 1999a]. Similarly, for the approach to be generic, it is important for the control of the behavioral model to be independent of the learning mechanism, so that the model might be piloted by any decisional mechanism.

The development of learning adds a whole extra dimension to our model (see figure 3.2). Indeed, our virtual entities evolve in a virtual world (first dimension: the virtual world), simulate the representation of behaviors in an imaginary world (second dimension: the imaginary world), and adapt the representation of behaviors through learning (third dimension: the abstract world).

The challenge here is therefore to identify the three dimensions and to understand their interactions. The three worlds evolve in parallel and correspond to three different levels of abstraction. Nevertheless, they are all related and share information. The virtual world provides the necessary information to the imaginary world in order to simulate an approximate representation of the virtual world. Furthermore, it provides the abstract world with the
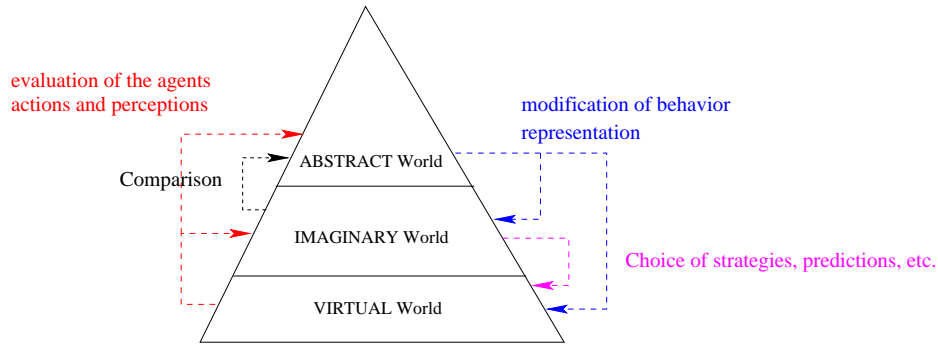
FIGURE 3.2 – Conceptual framework : representation of the entities' three dimensions (the real world, the imaginary world and the abstract world)

information it needs on the model, *i.e.* an approximation of these effectors and the sensors linked to the models to be adapted. The imaginary world feeds back information, particularly concerning the choice of strategies or predictions (figure 3.3).
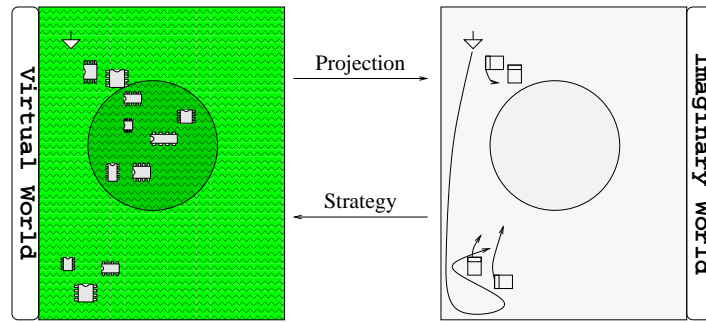
FIGURE 3.3 – Internal behavior simulation. The actor, represented by the triangle, establishes a simplified representation of the world. It simulates both its own behavior and that of the other entities in an imaginary world. After simulating a number of different possibilities, it can decide which strategy to adopt.

3.4 Application: JABU

One first implementation of this approach is a virtual juggler which predicts the displacement of balls in the air in order to coordinate its movements and juggle successfully. The application is called JABU: Juggler with Anticipatory Behavior in virtual Universe. The problem of virtual juggler was discussed by [Julliard and Gibet, 1999, Multon et al., 2001] but these approaches have not taken into account the modeling of generalized anticipation or the theory of simulation. More generally, they doesn't address the links between cognitive sciences and character's behavior. We will show that the conceptual framework proposed here can account for adaptation but also plausible errors, through interactions more or less predictable, especially with a real human. Applying the conceptual framework of anticipation for this example is shown in figure 3.4. The application is called JABU: Juggler with Anticipatory Behavior in

virtual Universe (see figure 3.5).
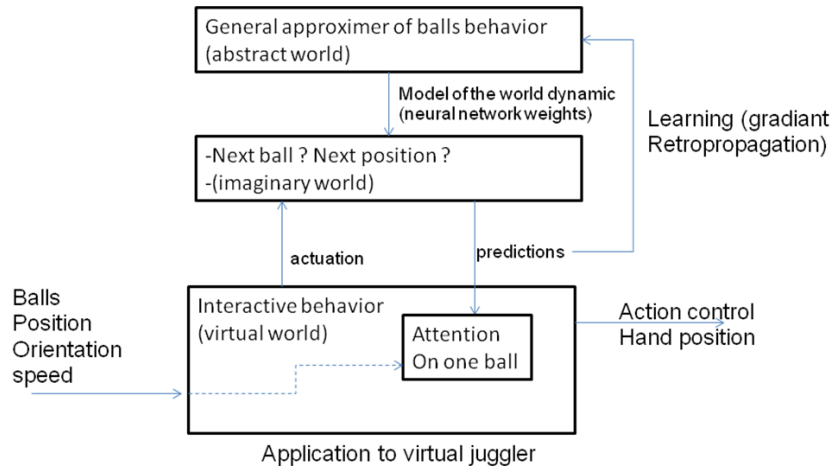
Figure 3.4 – Instantiation of our framework for a virtual juggler

The virtual world of the juggler has physical properties (inertia, gravity, wind, etc.) through the use of the ODE[3] physics engine. Of course, these quantities are not explicit in the model of control. This control is adjusted through an attentional process focused on the *next* (anticipated) ball (at this time one ball by hand). The approximate position of the balls is made by their simulation in the imaginary world of juggling. Function approximation properties of this imaginary world is coming from different neural networks. The abstract world corresponds to the weights of the arcs of these networks. Since they are universal approximators, we will see that they allow real-time adaptation of the juggler gestures to different types of disturbances. In the following, we clarify the implementation of these principles.

Figure 3.5 – Screenshots of the Jabu application.

---

[3] Open Dynamic Engine, http://www.ode.org/

3.4.1   Virtual world

3.4.1 - A   Presentation

The motor behavior of the juggler is controlled by his hands. The hands have independent functions, that is to say that there are no complex juggling moves or tricks, but simply a succession of catches and throws of the balls, where each movement is independent of the others. As soon as a ball "arrives" at the same height as the hands, it must be caught and rethrown. The time taken for a hand to move is not negligible and exposes the juggler to a risk of delay and thus "missing" the ball, which is also amplified by prediction errors. As mentioned above, the precise reproduction of the movement is not our priority and the hand's movement time is an empirically adjustable variable which reflects the delay between the decision being made and the action being carried out. In the following section, for simplicity's sake, and to keep things brief, when we refer to a hand's activity, we also of course mean that the theoretical model has been implemented for the anticipatory decision-making applied to our juggler.

The different phases of juggling are the following. The juggler begins by looking for a ball in the air. Once the ball is spotted, the hand must aim at an estimated reception point (prediction T1). Then, it is possible to refine this reception point. In order to do so, the hand must estimate and correct the anticipated trajectory of the target ball (prediction T2) which is the *object of attention*. Each hand will therefore be able to catch or miss the target ball. If the ball is caught, the juggler will be able to throw it in the air. Whether the ball is caught or missed, the hand again begins to look for a ball in the air.

3.4.1 - B   The link between the virtual and imaginary worlds

To aim at a reception point and to estimate the anticipated trajectory of the target ball, the hand will have to use predictive models. Within the context of juggling, information must be gathered quickly in order to maintain the juggling dynamic. The use of perceptron-type neural networks (NN) to make predictions about the trajectory is adequate. Indeed, NN are executed rapidly and online learning occurs both quickly and effectively. Furthermore, NN correspond to our need to manipulate (both spatial and temporal) digital data. It is, of course, also possible to use determinist equation models of movement to make predictions. However, such precise predictions would be extremely noise-sensitive (disruption of the environment as the ball falls) and would not account for the use of approximations and readjustments in real-time which seem to be the basis of the anticipatory mechanisms that we aim to respect [Berthoz, 1997].

It must also be noted that we are working from a pragmatic, rather than a neurophysiological perspective. In no way do we suggest that we are simulating "low-level" neural functioning like that in robotics [Laroque et al., 2010] but rather that we are creating an anticipatory behavior which is as effective as possible. Thus, perception must be seen as a simple approximation. It must also be noted that NN are in this case used as explicit models of anticipation rather than functioning.

We shall now go on to describe the predictive models used for simple juggling (juggle alone in section 3.4.2) and for juggling between a virtual entity and a human user (in section 3.4.3).

3.4.2   Imaginary world

This section describes the juggler's predictions T1 (section 3.4.2 - A) and T2 (section 3.4.2 - B).

3.4.2 - A   Prioritizing the balls (T1)

NN T1 will provide us with the estimated temporal and spatial data for each ball at the moment it is thrown. These data will be used to categorize the balls and attribute them priorities thus triggering the attentional process on the priority ball. The data required to calculate these estimations are the current speed of the ball and the height h at which the ball must be caught. This neural network includes:

▷ 3 inputs: the 3 speed components of the ball in 3D representing the 3 axis of the space

▷ 3 outputs:

  1. the estimated time (duration) before the ball crosses a plane in $z = h$ (h determined during learning)
  2. the movement in x, of the ball on crossing plane $z = h$
  3. the movement in y, of the ball on crossing plane $z = h$

x and y define the cartesian plane. The data used by the NN T1 are summarized in table 3.2. The information that it represents is illustrated in figure 3.6.

Inputs	Output	Parameter	Objectives
Vx	Δt	h	Temporal classification
Vy	Δx		Vague spacial prediction
Vz	Δy		

TABLE 3.2 – Input/Output of NN T1.

3.4.2 - B   Refining the prediction of the target ball (T2)

NN T2 refines the spatial prediction of where a ball will fall as it falls. Information can be obtained at different temporal levels (according to Δt). This neural network includes:

▷ 3 inputs: the 3 speed components of the ball in 3D

▷ 3 outputs: an estimation of the movement in x, y and z of the ball after a given time Δt (where Δt is defined during learning).

The data used by the NN T2 are summarized in table 3.3. The information that it represents is illustrated in figure 3.6.

Inputs	Output	Parameter	Objectives
Vx	Δx	Δt	Refined spatial predictions
Vy	Δy		
Vz	Δz		

TABLE 3.3 – Input/Output of NN T2.

Delta t

T1 (a)

T2 (b)
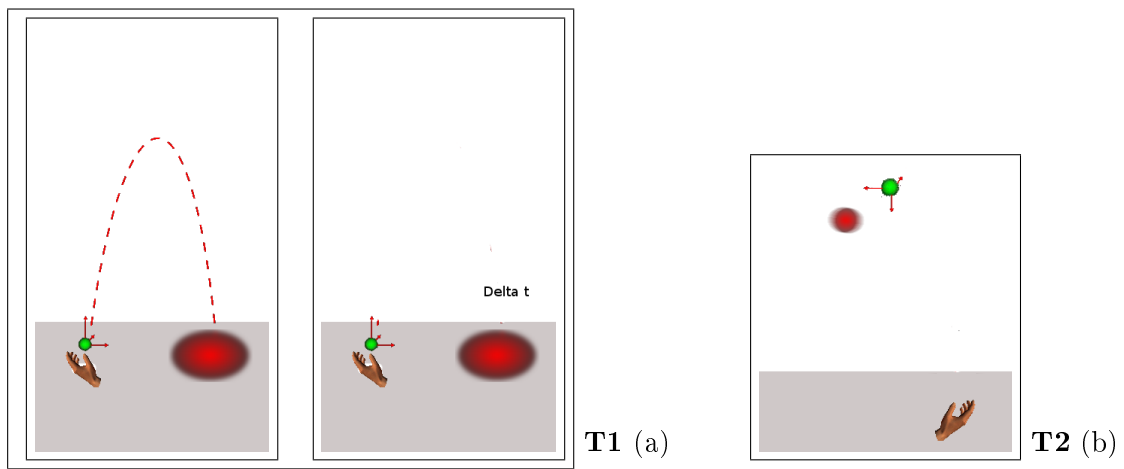
FIGURE 3.6 – **NN T1**: The ball has just been thrown at a speed in x,y,z (arrows), giving us a first estimate of the position at which it will cross plane $z = h$ (blurred circle) and how long this will take (a). **NN T2**: At any given time as the ball falls (ball with arrows), we can use its speed (arrows) to make a more accurate estimation of its position in Δt seconds (blurred ball) (b).

3.4.3 Interaction between virtual jugglers and with the human

The general features of this proposition allows several jugglers to interact together. To do that, the only change is the direction of the ball launched by each juggler (see figure 3.7 (a)).

Our juggler can also catch a new ball thrown by a human user (figure 3.7 (b)). This is extremely pertinent for evaluating the believability of our virtual juggler (real-time decision-making, online adaptation, etc.). Introducing a human user also requires the introduction of a new type of prediction (T3). T3 is similar to prediction T1, except that the ball is not thrown by the virtual juggler. The human user interacts with the virtual juggler using a Wiimote (remote games controller from the Nintendo Wii console). This peripheral device measures the movements of the human user's hand.

(a)

(b)
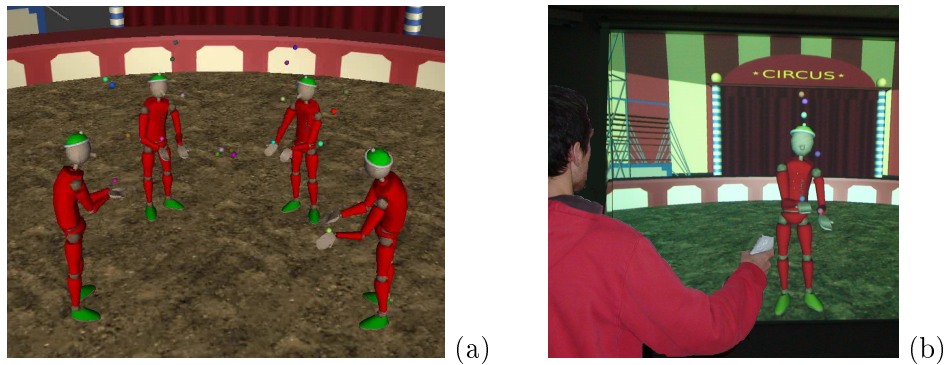
FIGURE 3.7 – Multi-jugglers (a) and a human can juggle with the virtual juggler using the Wiimote (b).

3.4.4   Abstract world

In its example base, NN T1 has access to throws made by the juggler itself (low speed in x and y) whereas NN T3 records the balls thrown at a distance by a third person (much greater speeds).

3.4.4 - A   Learning

We chose a topology with two hidden layers as the aim was to approximate a continuous function [Cybenko, 1989]. Each hidden layer has 19 neurons, and we thus obtain 3x19x19x3 multilayer perceptrons. We assign the perceptron weights with given values prior to learning. The activation function of the neurons is limited. The learning algorithm is a retropropagation of the gradient error. Learning is thus conducted with a maximum of 100 iterations using the FANN[4]. The parameters to be determined are h for the NN of T1 and Δt for the NN of T2. In our example, $h = 2.5cm$ and $\Delta t = 0.1s$.

3.4.4 - B   Verification

We divided the data into two subsets: the learning set and the validation set. The validation set is not used for learning but rather to verify the relevance of the network with unknown samples. In this case, we have a sample of 500 pairs of inputs/outputs for NN T1. The learning set uses 2/3 of this sample, and the rest is in the validation set. We obtain figure 3.8. The graph illustrates the mean quadratic error at each stage of learning for each sample. It must be noted that the learning set shows that the error decreases dramatically and the validation set confirms that overlearning does not occur.

---

[4] Fast Artificial Neural Network (FANN) library available at http://leenissen.dk/fann/
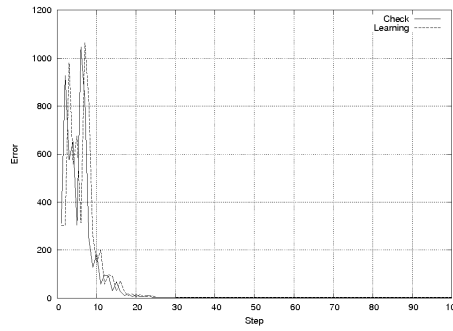
FIGURE 3.8 – Overall error in learning and validation

3.5 Evaluation: the effects of prediction on behavior, as applied to JABU

In this section, we shall evaluate the anticipatory mechanism, its qualities, and its impact on decision-making and the final result: the juggler animation. The generalization abilities of NN allows the in line adaption of the juggler's motion to disturbances. The tests presented will vary the initial conditions for a given time period. All of the tests include 42 balls thrown towards the virtual juggler (one ball every 0.75 seconds). We will observe the number of balls dropped by the juggler (*i.e.* which fall below the juggler's knees and which it is unable to catch).

3.5.1 Quality of the model

Here we will focus on the quality of our model enabling us to make predictions about T1 and T2.

First, we compared the performance of our juggler with the spatial and temporal predictions based on our NN and others based on equations of movement. We simultaneously launched ten $1min$ NN simulations and ten others using calculated equations. An average of 30 balls were dropped for the NN and 31 for the equations. We can thus conclude that the prediction of the NN T1/T2 and the equations (exact prediction) are equivalent, so NNs are good quality models for the juggling simulation.

We then attempted to distort the prediction model. To do so, we weighted the input/output data provided for learning according to a maximum error percentage. We compared the performance of our juggler for a distorted T1 prediction with an augmented maximum error percentage (from 0 - 180%) against the original data values. We conducted five simulations for each percentage. The experiment was based on $1min$ simulations for each maximum error percentage. We obtained the results presented in figure 3.9. The more the input/output data are distorted, the more balls the juggler drops. There is a distinguishable breaking point around the 120% maximum error. This therefore supports the reliability of our T1 prediction.
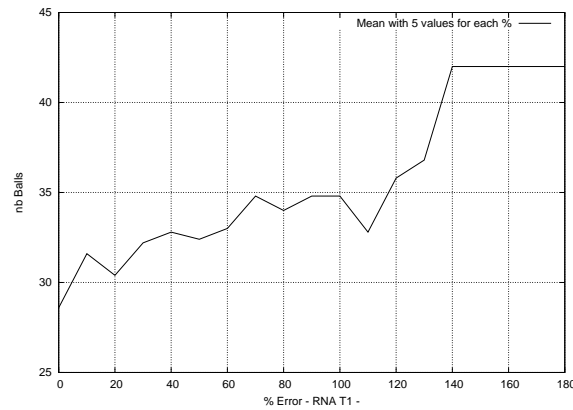
FIGURE 3.9 – Average number of balls dropped according to the error percentage of the data provided to the NN for the T1 prediction.

3.5.2 Disturbing the environment online

Here we experiment by distorting the model to validate its reliability. This will be tested by changing the projectile, varying gravity in the virtual environment and by adding wind (see figure 3.10). This information will not be given to the juggler; its imaginary world and its abstract world will therefore provide different conditions than the virtual world.

First, we introduce jerks in the projectile trajectories because they become maces rather than balls. In this case, the NN T1 is less precise in its prediction but the NN T2 is able to correct properly the prediction and the juggler continues to juggle when balls are *transformed* in maces.
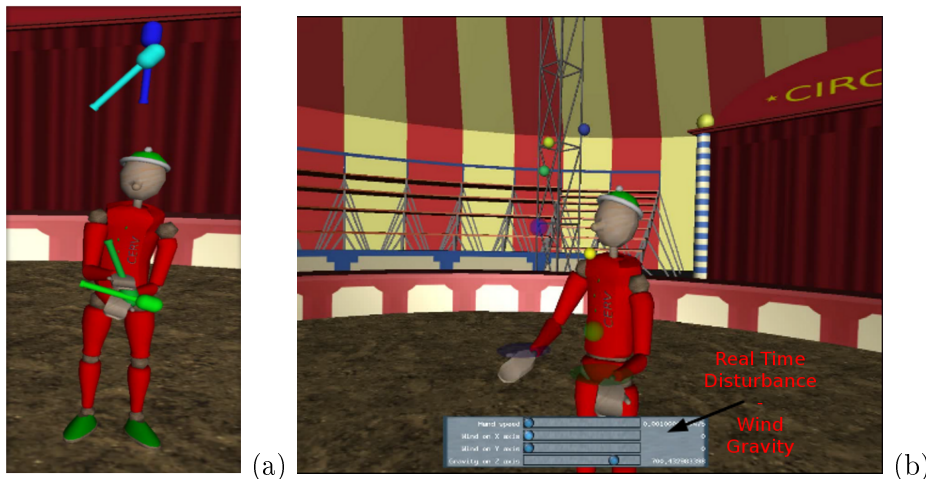
FIGURE 3.10 – Juggling with maces (a) and disturbing the environment conditions in line (wind, gravity) (b)

In figure 3.11, we introduced gravity variations. On the y axis the number of balls dropped, and in the x axis, the value of gravity in m/s^2. We calculated the mean for 10 values. The experiment was based on $1min$ simulations for each gravity value. We observed

that juggling was possible for gravitational values between 6 and 15 (normal gravity: 9.81). In cases of extremely low gravity, no balls were recorded as dropped, as they did not have the time to fall to the ground during the short simulation time.

In figure 3.12, wind was added. We thus obtain a curve for dropped balls according to wind in x (a) and in y (b). In the y axis we observe the number of balls dropped and in the x axis acceleration according to wind speed (in m/s^2, with direction indicated by positivity or negativity). The x axis value is an acceleration due to the fact that we use a modification of gravity to simulate the wind. The mean was taken for 5 simulations for each wind value.

For wind in x (width), we observe that between $-0.5m/s^2$ and $+0.5m/s^2$, the juggler catches most of the balls. Beyond that wind speed, it is much more difficult to juggle correctly. For wind in y (depth), the range of speeds in which the juggler continues to juggle correctly is much smaller (between $-0.2m/s^2$ and $+0.2m/s^2$).
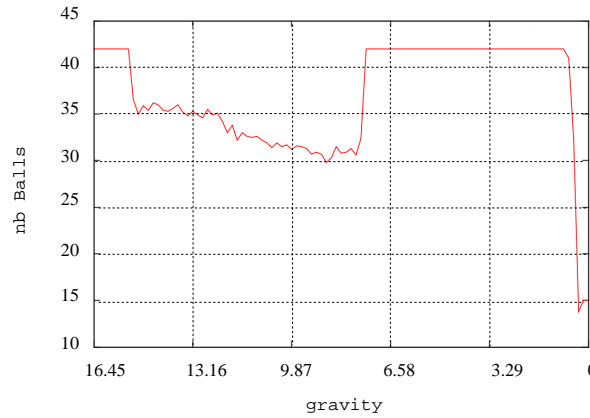
FIGURE 3.11 – Average number of balls dropped according to gravity.
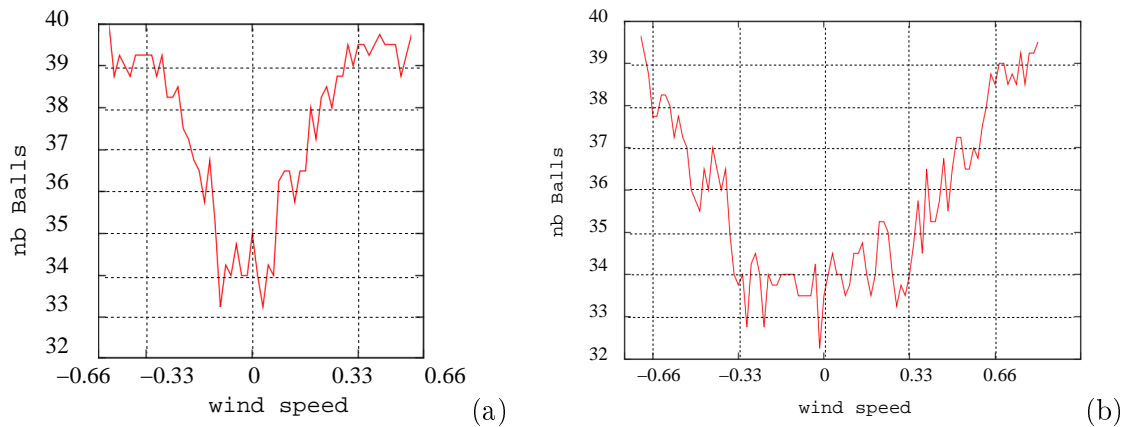
FIGURE 3.12 – Mean number of dropped balls according to wind speed in y (a) and in x (b).

3.5.3 Relationship with decision-making

The predictive model provides information for decision-making; here, the choice of hand movements. We experiment with variations in the execution of the decision-making model for one given predictive model. We decided to vary hand speed. Figure 3.13 illustrates the evolution of the average number of dropped balls according to varying hand movement speed. For this experiment, we conducted 19 simulations at the same speed. The change in speed took place at $0.0005s$ intervals. The speed boundaries varied from $0.0005s$ to $0.1s$ and the experimental simulation lasted $2min$.
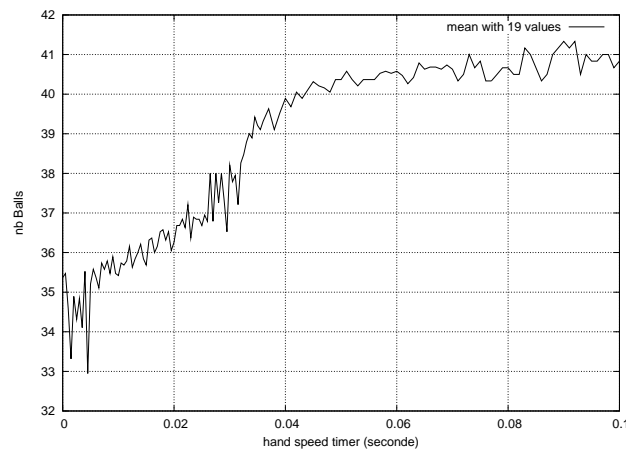
FIGURE 3.13 – Average number of balls dropped according to hand speed.

It was observed that beyond $0.04s$ a high percentage of balls were dropped. Despite the anticipatory mechanism, decision-making no longer enabled the juggler to juggle successfully. We also noted that the juggler could no longer juggle with more than 9 balls at once.

3.6 Conclusion

To increase the behavioral believability of the interaction of a virtual entity, it would seem essential to integrate an anticipatory capacity by which the behavior of other entities and their consequences on the environment can be predicted. In order to do so, we suggest an architecture by which the three modes - reactivity, predictability and adaptability - can function asynchronously in parallel. The prediction is made by an autonomous world of *simulation within the simulation*, in which the entity can simulate itself (with its own behavioral model) and its environment (with the representations which it constructs of the behaviors of other entities). Our work takes a pragmatic approach, far removed from neuromimetics, with a model which integrates an anticipation principle, notably with the possibility of prediction errors.

We developed a virtual juggler that anticipates the trajectory of the balls without calculating them precisely. Indeed the juggler hypothesizes using an open and uncertain environment with variable properties, that is to say, that are unknown from an analytical standpoint.

We therefore use universal approximators obtained through learning.

Two types of predictions offer answers to the questions of which will be the next ball to catch and where it will fall. Another prediction accounts for the balls thrown by human user using a Wiimote. These predictions are made and refined by neural networks. Through our application, known as Jabu, we were able to evaluate our proposal using a number of experiments.

Of course, this work don't address the quality of gestures, nor the comparison with real data from juggling. To do that, we have in perspective the improvement of this proposition with realistic models of gesture. For the moment, the purpose was to show that it is possible to exhibit plausible failures in the task when taking into account simulation and anticipation. Another important point is that the juggler is able to juggle with human which take part of an unpredictable environment.

We are currently orienting our work toward the addition of different juggling strategies. The imaginary world of a simulation within a simulation could be used to test many different possibilities. The results of such simulations would help to provide strategies which are better adapted to the virtual world.

We would also like to work on a new kind of prediction dealing with the behavior of the human interacting with the juggler. In the current application the interaction between the human and the virtual juggler occurs using a Wiimote. This peripheral device measures the movements of the human user's hand. The virtual juggler has access to this data, thus enabling it to "watch" the user. A recognition mechanism could thus be conducted by the juggler using the observed data, in order to identify information which could offer clues about the human user's future behavior. In this way, the juggler could recognize a movement which it identifies as potentially the start of a throw (the user's arm moves backwards to build up speed to throw the ball to the juggler). This movement to build up speed will provide the juggler with information about how imminent the throw is as well as the probable trajectory of the ball, so that it might anticipate the ball and react accordingly. Here, this might mean paying more attention to the user, throwing the balls which are currently being juggled higher, etc.

Contents of Chapter 4

Chapter 4

CAMUS — fuzzy cognitive maps for the simulation of individual behaviors

This chapter focuses on the simulation of behavior for autonomous entities in virtual environments. The behavior of these entities must determine their responses not only to external stimuli, but also with regard to internal states. We propose to describe such behavior using fuzzy cognitive maps (FCM), whereby these internal states might be explicitly represented. This chapter presents the use of fuzzy cognitive maps as a tool to specify and control the behavior of individual agents. First, we describe how fuzzy cognitive maps can be used to model behavior. We then present a learning algorithm allowing the adaptation of FCMs through observation.

4.1 Introduction

FOR simulation purposes, decision making by autonomous entities is defined according not only to external stimuli, but also to internal states. In this chapter, we show that such behaviors can be described using fuzzy cognitive maps (FCMs) in which these internal states will be explicitly represented. The strengths of FCMs lie in the fact that they can be used to graphically represent specific behavior in the form of a semantic graph and that their evaluation at run-time is fast enough to meet the requirements of real-time simulations, as do connectionist architectures.

FCMs are the outcome of research by psychologists. In 1948, Tolman introduced the key concept of "cognitive maps" to describe complex topological memorizing behavior in rats [Tolman, 1948]. In the seventies, Axelrod described "cognitive maps" as directed, interconnected, bilevel-valued graphs, and used them in decision theory applied to the political-economics field [Axelrod, 1976]. In 1986, Kosko extended the graphs of Axelrod to the fuzzy mode which thus became FCM [Kosko, 1986b]. In 1994, Dickerson and Kosko proposed the use of FCMs to obtain overall virtual world modeling [Dickerson and Kosko, 1994]. Recently, FCMs have been successfully used to describe and model complex dynamic systems

[Koulouriotis et al., 2003], both for medical diagnosis [Stylios et al., 2008] and in decision-making [Rodriguez-Repiso et al., 2007].

In all these studies, FCMs have been used to control a global system. Here, we propose to decentralize FCMs onto each agent, in order to model the agents' decisions within a virtual world. This chapter proposes the use of FCMs as a tool to model the reactive and adaptive behavior of agents improvising in free interaction.

The chapter is organized as follows. First, we highlight the fact that FCMs are particularly well adapted to specifying and controlling agents' decisions. We present the uses of FCMs in reactive behavior, illustrating these uses with a real-life example involving different types of agent: a shepherd, dogs and a herd of sheep. Second, we describe the ability provided for agents to adapt their representation of other actors' behavior using FCMs, which leads to their predictions becoming more significant. This means that we give an agent the ability to learn through imitation. The agent is then able to modify its own behavior to mimic an observed behavior by either another actor or an avatar controlled by a human operator. By observing the imitated model, the agent must adapt its representation of the model behavior. The mechanism used to control the imitating behavior model is independent of learning. Thus, imitated models can be driven by any decision-making mechanism. We apply this algorithm to the example given above (a sheepdog herding sheep). The learning mechanism allows the dog to adapt an FCM prey prototype to a given sheep in real time. The project is called CAMUS for Cognitive and Adaptive Map for Unsupervised Simulation.

4.2 Reactive behavior with Fuzzy Cognitive Maps

4.2.1 FCM presentation

FCMs are influence graphs (see fig 4.1). Nodes are named by concepts forming the set of concepts $\mathcal{C} = \{C_1, \cdots, C_n\}$. Edges (C_i, C_j) are oriented and represent the causal links between concepts (how concept C_i causes concept C_j). Edges are elements of the set $\mathcal{A} = \{(C_i, C_j)_{ij})\} \subset \mathcal{C} \times \mathcal{C}$. The edges' weights are associated with a link value matrix: $L_{ij} \in M_n(\mathbb{R})$. If $(C_i, C_j) \notin \mathcal{A}$ then $L_{ij} = 0$, else the excitation link (and inhibition link respectively) from concept C_i to concept C_j gives $L_{ij} > 0$ ($L_{ij} < 0$ respectively).

FCM concept activations take their value from an activation degree set $\mathcal{V} = \{0, 1\}$ or $\{-1, 0, 1\}$, or an interval. At moment $t \in \mathbb{N}$, each concept C_i is associated with two types of activations: an internal activation degree $a_i(t) \in \mathcal{V}$ and an external forced activation value $f_{a_i}(t) \in \mathbb{R}$. $a(0) = \mathbf{0}$, where $\mathbf{0}$ is the $\mathbb{R}^n$ null vector.

FCMs are dynamic systems. The dynamic obeys a recurring relationship involving link matrix products with internal activation vectors, and fuzzy logical operators between this result and external forced activation vectors.
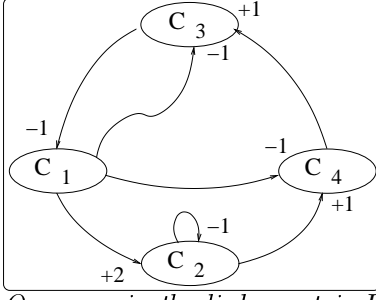
The FCM seen opposite is made up of 4 concepts and has 7 edges. At a moment t, each concept C_i has a degree of activation $a_i(t)$. The activation $a(t) \in \mathcal{V}^4$ and the links $L \in \mathcal{M}_4(\mathbb{R})$ are:

$$a(t) = \begin{pmatrix} a_1(t) \\ a_2(t) \\ a_3(t) \\ a_4(t) \end{pmatrix} \qquad L = \begin{pmatrix} 0 & +2 & -1 & -1 \\ 0 & -1 & 0 & +1 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & +1 & 0 \end{pmatrix}.$$

One zero in the links matrix $L_{ij} = 0$ indicates the absence of edges from concept C_i to concept C_j, and a diagonal element $L_{ii} \neq 0$ corresponds to an edge from concept C_i to itself.

FIGURE 4.1 – An FCM as an influence graph.

4.2.2 Formalization of the FCM dynamic

Until the end of this chapter, δ indicates one of the numbers 0 or 1, and $\mathcal{V}$ one of the sets $\{0, 1\}$, $\{-1, 0, 1\}$ or $[-\delta, 1]$. Given $n \in \mathbb{R}^*$, $t_0 \in \mathbb{N}$, $\rho \in \mathbb{R}_+^*$, and $a_0 \in \mathbb{R}$.

A fuzzy cognitive map $\mathcal{F}$ is a sextuplet $< \mathcal{C}, \mathcal{A}, L, A, f_a, \mathcal{R} >$ where:

1. $\mathcal{C} = \{C_1, \cdots, C_n\}$ is the set of n concepts forming the nodes of a graph.

2. $\mathcal{A} \subset \mathcal{C} \times \mathcal{C}$ is the set of edges (C_i, C_j) directed from C_i to C_j.

3. $L : \left| \begin{array}{ccc} \mathcal{C} \times \mathcal{C} & \to & \mathbb{R} \\ (C_i,\ C_j) & \mapsto & L_{ij} \end{array} \right.$ is a function $\mathcal{C} \times \mathcal{C}$ to $\mathbb{R}$ associating a weight L_{ij} to a pair of concepts $(C_i,\ C_j)$, with $L_{ij} = 0$ if $(C_i,\ C_j) \notin \mathcal{A}$, or with L_{ij} equal to the weight of the edge directed from C_i to C_j if $(C_i,\ C_j) \in \mathcal{A}$. $L(\mathcal{C} \times \mathcal{C}) = (L_{ij}) \in \mathbb{R}^{n \times n}$ is a matrix of $\mathcal{M}_n(\mathbb{R})$. It is the link matrix of the map $\mathcal{F}$ that, to simplify, we note L unless indicated otherwise.

4. $A : \left| \begin{array}{ccc} \mathcal{C} & \to & \mathcal{V}^{\mathbb{N}} \\ C_i & \mapsto & a_i \end{array} \right.$ is a function that maps each concept C_i to the sequence of its activation degrees such as for $t \in \mathbb{N}$, $a_i(t) \in \mathcal{V}$ is its activation degree at the moment t. We note $a(t) = [(a_i(t))_{i \in [\![1,n]\!]}]^T$ the vector of activations at the moment t.

5. $f_a \in (\mathbb{R}^n)^{\mathbb{N}}$ is a sequence of vectors of forced activations such as for $i \in [\![1, n]\!]$ and $t \geq t_0$, $f_{a_i}(t)$ is the forced activation of concept C_i at the moment t.

6. $(\mathcal{R})$ is a recurring relationship on $t \geq t_0$ between $a_i(t+1)$, $a_i(t)$ and $f_{a_i}(t)$ for $i \in [\![1, n]\!]$ indicating the dynamics of the map $\mathcal{F}$.

$$(\mathcal{R}) : \forall i \in [\![1, n]\!],\ \forall t \geq t_0, \begin{cases} a_i(t_0) = 0 \\ a_i(t+1) = \sigma \left[g_i \left(f_{a_i}(t), \sum_{j \in [\![1,n]\!]} L_{ji} a_j(t) \right) \right] \end{cases} \tag{4.1}$$

where $g_i : \mathbb{R}^2 \to \mathbb{R}$ is an operator combining two variables, for example:

$$g_i(x, y) = \min(x, y)\ ,\ \text{or}\ \max(x, y)\ ,\ \text{or}\ \alpha_i x + \beta_i y\ ,\ \ldots$$

and where $\sigma : \mathbb{R} \to \mathcal{V}$ is a function from $\mathbb{R}$ to the set of activation degrees $\mathcal{V}$ normalizing the activations as follows (see fig 4.2):

(a) If continuous mode (called fuzzy mode), $\mathcal{V} = [-\delta, 1]$, σ is the sigmoid function $\sigma_{(\delta, a_0, \rho)} : a \mapsto \frac{1+\delta}{1+e^{-\rho(a-a_0)}} - \delta$ centered in $(a_0, \frac{1-\delta}{2})$, with a slope $\rho \cdot \frac{1+\delta}{2}$ in a_0 and with limits at $\pm\infty$ respectively 1 and $-\delta$. The larger ρ is, the less linear the transformation will be. In practice, $\delta \in \{0, 1\}$: 0 stands for a bivalued discret logic or a fuzzy logic with values in $[0, 1]$, while 1 corresponds to trivalued discret logic or a fuzzy logic with values in $[-1, 1]$.

(b) If bimodal mode, $\mathcal{V} = \{0, 1\}$, $\sigma : a \mapsto \left| \begin{array}{l} 0 \text{ if } \sigma_{(0, 0.5, \rho)}(a) \leq 0.5 \\ 1 \text{ if } \sigma_{(0, 0.5, \rho)}(a) > 0.5 \end{array} \right.$.

(c) If ternary mode, $\mathcal{V} = \{-1, 0, 1\}$, $\sigma : a \mapsto \left| \begin{array}{lll} -1 & \text{if} & \sigma_{(1, 0, \rho)}(a) \leq -0.5 \\ 0 & \text{if} & -0.5 < \sigma_{(1, 0, \rho)}(a) \leq 0.5 \\ 1 & \text{if} & \sigma_{(1, 0, \rho)}(a) > 0.5 \end{array} \right.$.

The asymptotic behavior $(t \to +\infty)$ of FCMs with constant externally-forced activation vector sequence may be a fixed point, a limit cycle, or even a strange attractor if it is sufficiently complex [Kosko, 1986b].
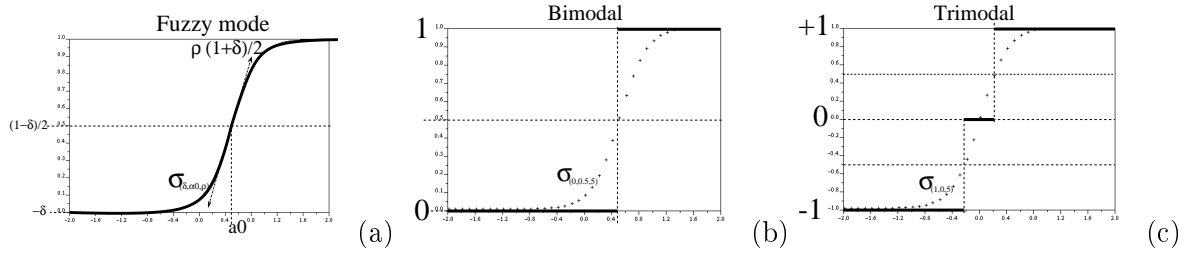
FIGURE 4.2 – Cognitive maps' standardizing functions

4.2.3 FCM for which behavior?

It is difficult to describe the entire behavior of a complex system with a precise mathematical model [Hafner, 2000]. It is therefore more useful to represent it graphically, showing the causal relationships between the concepts involved. Therefore, FCM can avoid many of the knowledge-extraction problems which are usually present in rule based systems [Stylios et al., 1997b].

FCMs are capable of forward chaining only, *i.e.* they can answer the question "What would happen if...?", but not "Why...?", due to the non-linear nature of FCM dynamics. FCMs help predict the system's evolution (behavioral simulation) and can be equipped with Hebbian learning capabilities, as proposed by Dickerson and Kosko [Dickerson and Kosko, 1994]. The fundamental difference between an FCM and a Neural Network (NN) is that, in FCM, all nodes have strong semantics defined by the concept's model, whereas internal nodes in NN graphs have weak semantics which are only defined by mathematical relationships. As regards the learning capacity, during the training phase, activation vectors must be given for all concepts of FCMs, whereas for NNs, activation vectors are only required for the peripheral neurons.

4.2.4 Constructing FCM

FCMs use symbolic representations which are able to incorporate experts' knowledge [Hagiwara, 1992, Stylios and Groumpos, 2000, Salmeron, 2009]. Human experts have drawn up FCMs for planning or making decisions in the field of international relations and political developments [Taber, 1991], to model intruder detection systems [Siraj et al., 2001] and demonstrate the impact of drug addiction [Calais, 2008].

First of all, the experts determine the concepts that best describe the system. They know which factors are crucial for modeling the system (characteristic, state, variable, etc.) and provide a concept for each one. Next, they identify those elements of the system which influence other elements and, for the corresponding concepts, determine the positive or negative effect of one concept on the others. Finally, not all causes affect a factor with the same intensity. Some of them have a greater, and others a lesser effect. The experts must assign effect intensities. In order to simplify matters, they might separate the relationships between factors into groups, for example high intensity ($L_{ij} = 6$), moderate intensity ($L_{ij} = 3$), and low intensity ($L_{ij} = 1$).

Within the framework of studies to design an autopilot for racing yachts, we proposed to model the cognitive activity of action selection using FCMs [Parenthoën et al., 2002b]. In such FCMs, concepts are action strategies or affordances. Originally introduced by Gibson in ecological psychology, an affordance is defined as an "action possibility" latent in the environment, objectively measurable and independent of the individual's ability to recognize it, but always in relation to the actor [Gibson, 1979]. In collaboration with an ergonomist, an expert lists a set of required affordances related to the activity, which is to be modeled. The affordance approach needs a model which explain how an individual selects one affordance out of a few; this where we use FCMs. Rather than considering activation levels or the internal inhibition of action graphs based on releases [Norman and Shallice, 1986], we again worked from the notions of attractors and repulsors in external environments, as suggested by research on affordance selection [Lewin, 1936, Reed, 1993, Lahlou, 2007]. From this point of view, an affordance is not necessary an actual invariant of the environment, but it is a hypothesis made by the agent based on its immediate environment, which is associated with an action strategy. One part of the expert's knowledge is translated into inhibition/excitation relations between affordances: for instance, obstacles could inhibit pathways, gateways and landmarks, while gateways inhibit each other. This gives the link matrix (L_{ij}). The other part of the expert's knowledge concerns the affordance perception value. As proposed in experimental psychology [Stoffregen et al., 1999a], for each affordance concept the expert proposes a formula for this perception value, which we use as the external activation f_{a_i} of the affordance concept. The FCM dynamics occur and activations a_i converge towards its attractor. The selected affordance i is the greatest $a_i(t)$ while $a(t)$ follows the path of the attractor (most often a fixed point or a limit cycle). Such a virtual agent acts according to the expert description and then increases its credibility [Mateas, 1997].

We have also used FCM to model emotional states. In collaboration with psychologists, we have described Fuzzy Emotional Map (FEM) model, first presented in [Nédélec et al., 2005]. Each emotion is modeled as an FCM. In this model we defined sensitive concepts (emotion input and state of mind), one output concept to determine an emotional intensity and four internal concepts to represent perception, feeling, sensitivity and the construction of emo-

tional ouput. The only features that require modification between different types of FEM are the influence weights between concepts of the map. Each weight is defined by a particular personality trait (*e.g.* anxiety or relaxation), and is used according to a specific kind of emotion.

4.2.5 FCM for modeling reactive agents' decision making

4.2.5 - A Principle

FCMs can specify and control the behavior of autonomous entities (agents). Agents have sensors and effectors, and make independent decisions with respect to their behavior. FCMs working in relation with these agents have perceptive, motor and internal concepts. The agents' decision-making is replaced by FCM dynamics in this way:

▷ agents' sensors define FCM perceptive concept activations through fuzzification[1]

▷ defuzzification[2] of FCM motor concept activations provides agents' effectors.

Fuzzification and defuzzification are obtained using the principles of fuzzy logics [Wenstop, 1976, Kosko, 1992], where a specific concept is represented by a fuzzy subset, and its degree of activation represents the degree to which it belongs to this subset [Sugeno and Nishida, 1985] (calculated using the membership function of the fuzzy set).

As an example, we aim to model an agent capable of perceiving its distance from an enemy. The agent will decide whether or not to escape from the situation, depending on this perceived distance. The closer the enemy is to the agent, the more frightened it will be, and *vice-versa*. The more frightened the agent, the more quickly it will try to flee. We model this escape behavior using the FCM in Figure 4.3a. This FCM has 4 concepts and 3 links: "enemy close", "enemy far", "fear" and "escape", with stimulating links ($+1$) from "enemy close" to "fear" and from "fear" to "escape", and an inhibiting link (-1) from "enemy far" to "fear". We chose fuzzy mode ($\mathcal{V} = [0, 1], \delta = 0, \rho = 5, a_0 = 0.5$), not forced ($f_a = 0$). The sensitive concepts "enemy close" and "enemy far" are activated by the fuzzication of the sensor for the distance to the enemy (Figure 4.3c) while the defuzzification of "escape" gives this agent an escape speed (Figure 4.3d).

Sensation must be distinguished from perception, in that sensation results from the sensors alone, whereas perception is the sensation influenced by an internal state. FCMs make it possible to model perception, thanks to the links between central concepts and sensitive concepts. For example, let us add 3 links to the previous escape FCM (Figure 4.3b). An initial self-stimulation of "fear" (link from "fear" to "fear" with ($\gamma \geq 0$)) simulates the effect of "stress": the more afraid the agent is, the more afraid it will feel. A second stimulation

---

[1] Fuzzification consists in converting external FCM values to FCM concept activations. fuzzification is a function from $\mathbb{R}^n$ to $\mathcal{V}$.

[2] Defuzzification consists in converting FCM concept activation to FCM external values. Defuzzification is a function from $\mathcal{V}$ to $\mathbb{R}^n$.
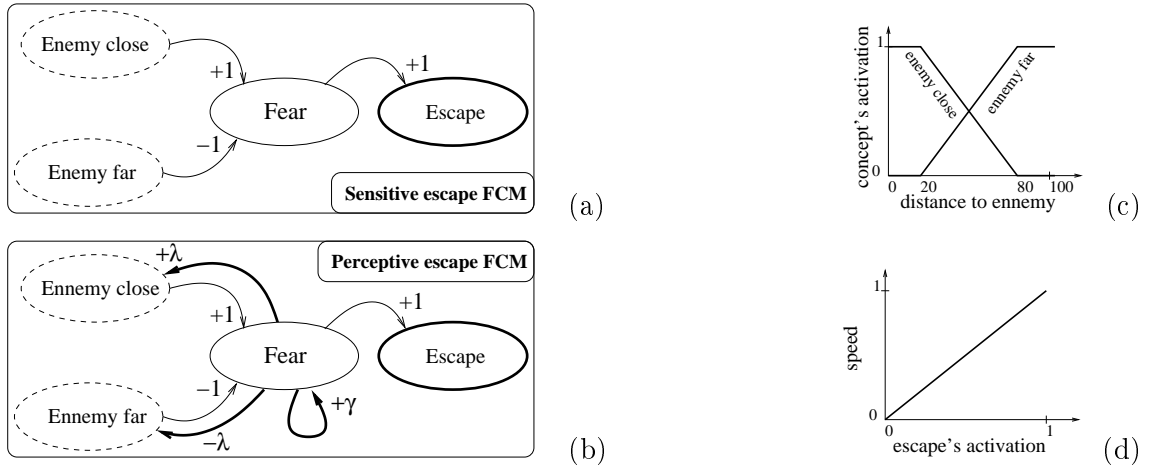
The sensitive concepts surrounded by dashes are activated by the fuzzification of sensors. The motor concepts in thick black lines activate the effector by defuzzyfication. In (a), the concept C_1 ="enemy close" excites C_3 ="fear" whereas C_2 ="enemy far" inhibits it and "fear" excites C_4 ="escape". A purely sensitive FCM is hereby defined. In (b), the FCM is perceptive: "fear" can be self-maintained (memory) and even influence feelings (perception). In (c), the fuzzification of the distance to an enemy gives two sensitive concepts: "enemy close" and "enemy far". In (d), the defuzzification of "escape" governs the speed of escape in a linear manner.

FIGURE 4.3 – FCM for an agent's escape behavior.

($\lambda \geq 0$) goes from "fear" to "enemy close" while a final inhibitor ($-\lambda \leq 0$) from "fear" to "enemy far" simulates the phenomenon of being "fearful", *i.e.* when the agent is afraid, it tends to perceive its enemy as being closer than it actually is. The agent becomes perceptive according to its degree of fearfulness λ and stress γ (see Figure 4.4).

4.2.5 - B   Application

This section illustrates the usage of FCMs in simulating the behavior of believable agents. In this example, FCMs characterize believable agent roles in interactive fictions through a story taking place in a mountain pasture. "*Once upon a time there was a shepherd, his dog and his herd of sheep . . .*" This example has already been used as a metaphor for complex collective behaviors within a group of mobile robots (RoboShepherd [Schultz et al., 1996a]), as an example of a watchdog robot for real geese (Sheepdog Robot [Vaughan et al., 2000]), and as an example of improvisation scenes (Black Sheep [Klesen et al., 2000]).

The shepherd moves around in the pasture and can talk to his dog and give it information. He wants to round up his sheep in a given area. In this simulation, the shepherd is an avatar for a human actor that makes all his decisions. Thus, no FCM is attached to this actor. By default, he remains seated.

Each sheep can distinguish an enemy (a dog or a human) from another sheep and from edible grass. It can evaluate the distance and the relative direction (left or right) from an agent in its field of vision. It is able to identify the closest enemy. It can turn left or right and
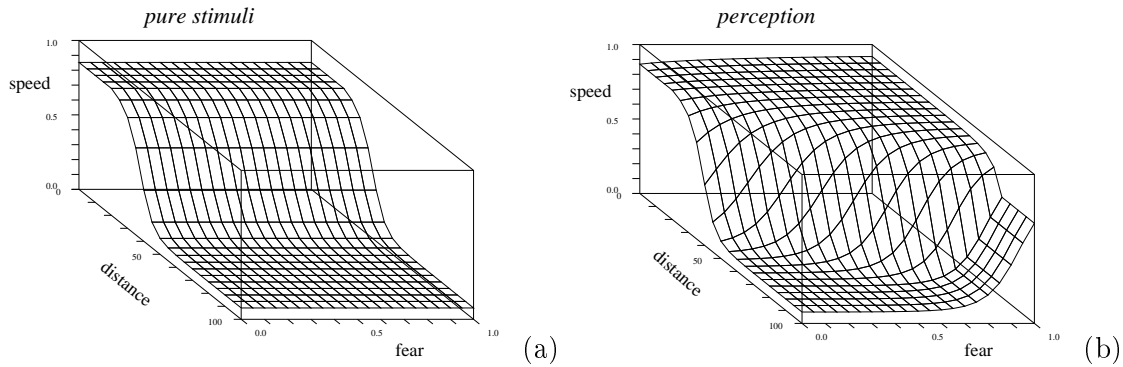
The perception of the distance to an enemy can be influenced by fear: depending on both the proximity of an enemy and fear, the dynamics of the FCM decide upon a speed obtained here with its 3rd cycle. In (a), $\lambda = \gamma = 0$, the agent is purely sensitive and its perception of the distance to the enemy does not depend on fear. In (b), $\lambda = \gamma = 0.6$, the agent is perceptive: its perception of the distance to an enemy is modified by fear.

FIGURE 4.4 – Escape speed decided by Figure 4.3b FCM

run without exceeding a certain maximum speed. It has an energy reserve that it regenerates by eating and exhausts by running. By default, it moves in a straight line and ends up wearing itself out. We want the sheep to eat grass (random response), to be afraid of dogs and humans when they are too close, and, in keeping with their gregarious nature, to stay close to other sheep. So, we chose a main FCM containing sensory concepts (enemy close, enemy far, high energy, low energy), motor concepts (eat, socialize, flee, run) and internal concepts (satisfaction, fear). This FCM calculates moving speed through defuzzification of the concept "run", and the direction of movement by defuzzification of the three concepts "eat", "socialize" and "flee". Each activation corresponds to a weighting on the relative direction to be followed: to go towards the grass, join another sheep, or to flee from an enemy respectively.

The dog is able to identify humans, sheep, the specific herding area within the pasture and the guard point. It distinguishes its shepherd from other humans and knows how to spot the sheep that is the farthest away from the area among a group of sheep. It knows how to turn to the left and to the right and run up to a maximum speed. Its behavior consists in running after the sheep, which quickly scatters them (see Figure 4.5a).

First, the shepherd wants the dog to obey the order "stay", which will lead the sheep to socialize. This is done by giving the dog a sensory FCM of the shepherd's message, which inhibits the dog's desire to run (see Figure 4.5b). The dog's behavior is driven by the FCM and the dog keeps still when asked to do so (message "stop"). The dog has an FCM based on the concepts associated with the herding area, for example whether a sheep is either inside or outside the area. These concepts also make it possible for the dog to bring a sheep back (Figure 4.6cde) and keep the herd in the area by staying at the guard point, in other words, on the perimeter of the herding area and opposite the shepherd.

It is remarkable to observe the virtual sheepdog's path in this simulation forms an S shape (Figure 4.6c), which is a strategy that can be observed for real sheepdogs rounding up sheep.
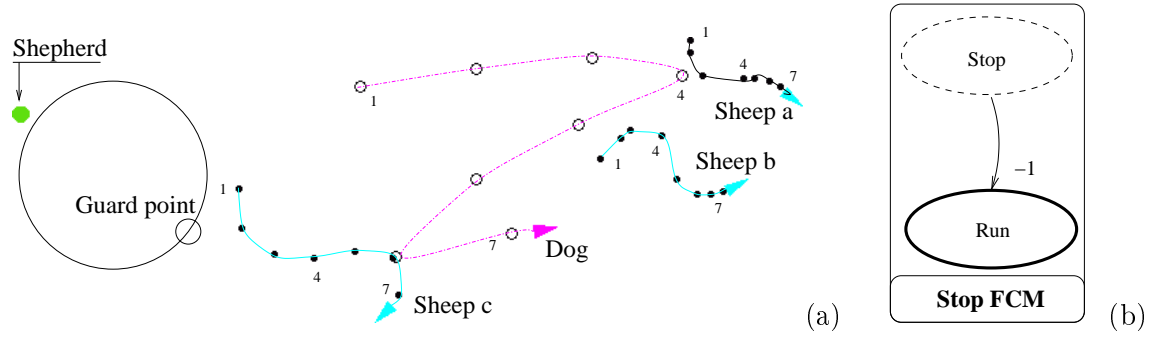
In (a), the shepherd is motionless. A circular zone represents the area where the sheepdog must gather the herd. A guard point is located in the zone, diagonally opposite the shepherd. The sheepdog must return to this point when all the sheep are inside the designated area. Initially, the behavior of a dog without FCM is to run after the sheep which then quickly disperse outside of the herding area. In (b), this elementary FCM carries out the actions of a dog obeying the shepherd's order to "stay", by inhibiting the desire to run.

FIGURE 4.5 – Sheepdog and sheep playing roles given by their FCMs.

4.3 Adaptive behavior with Fuzzy Cognitive Maps

To obtain believable behavioral simulations, agents of the same type must have slightly differents behaviors, and these individual behaviors must evolve over time. The actual behavior of a given agent is the result of its adaptation to the situations it has encountered. As each agent has its own past, evolution induces individual variability amongst agents' behaviors. When interacting with other agents, one agent has to adapt its behavior according to the way the behavior of its protagonists develops. For example, in a prey-predator interaction, the co-evolution of the two species has been observed in many cases and is known as the "Red Queen Effect" [Van Valen, 1973].

4.3.1 Imitation from prototypic behavior

The idea is to provide the ability for the agent to adapt its representation of other actors' behavior. This learning is done using the comparison between the simulation model and the observation of reality [Schultz et al., 1996a]. We propose a learning based on imitation [Meltzoff, 1995, Gallese, 2000] of observed behaviors to modify predefined prototypes.

Four main types of approach to learning by imitation can be distinguished: logical, connectionist, probabilistic and prototypical approaches.

1. *Logical approach.* Learning consists in generating a set of rules based on logic [Isla, 2001, Yu and Terzopoulos, 2007] and the sensorimotor observation describes the example (see XSTL logic [Del Bimbo and Vicario, 1995]). This approach is difficult to adapt to our perceptual modeling behavior based on FCM, requiring that the weighting of the edges' be linked with such sensorimotor rules.
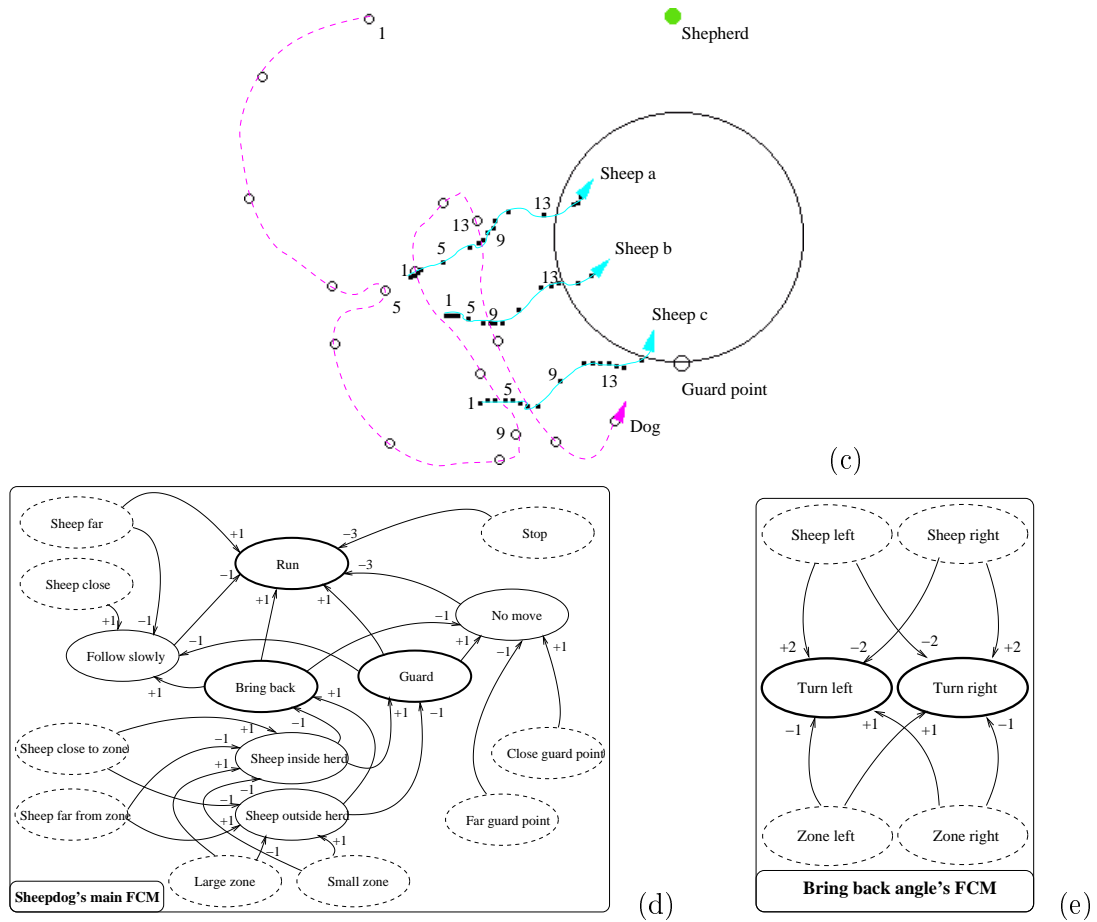
For this simulation, the sheep's desire to gather is inhibited. In (c), the film of the sheepdog bringing back three sheep was paused. The FCMs for the sheep and for the dog are represented in (d) and (e) respectively. In (d), the dog's main FCM represents the role of bringing a sheep back to the area and maintaining a position in relation to the shepherd when all the sheep are in the desired zone. In (e), this FCM decides the angle of incidence towards the sheep to bring it back into the zone: to go towards the sheep, but to approach it from the opposite direction.

FIGURE 4.6 – Sheepdog and sheep carrying out the roles given by their respective FCMs.

2. *Connectionist approach.* The actions are correlated with sensation by an adaptive neural network, possibly inhibited by a mechanism of exception recognition (see: architecture PerArc [Gaussier et al., 1998]); modeling a neural network provides a statistically satisfactory, but not semantically explicit, behavior instead of FCMs.

3. *Probabilistic approach.* Many internal variables are used [Le Hy et al., 2004], but since they do not model the emotions and do not reflect our concept of perception, these internal variables do not change the variables through sensory feedback effects.

4. *Prototypical approach.* Learning from prototypes creates an animation by finding primitives generating the imitated movement [Voyles et al., 1997, Mataric, 2002]. An FCM is an explanatory model suited to behavior specification. Thus, an expert will be able to develop a library of prototypic behaviors. This library represents the agent's behavioral culture [Mataric, 2002]. For example an animal's library is made up of the prototypic behavior of both predator and prey. Therefore, our agents have a library of prototypical behaviors, but unlike Mataric or Voyles, our primitives are at the decision of the movements, not within the movements themselves.

4.3.2 Principle

We consider that an agent has sensors allowing it to perceive its environment, as well as the effectors it uses to perform action. Any given agent also has a library of prototypic behaviors specified by FCMs.

In parallel to the virtual world, an agent also has an imaginary world, in which it can simulate its own behavior as well as the behavior of other actors. This imaginary world corresponds to an approximate representation of the environment from the agent's point of view, along with the representation of other actors' behavior. Agents use prototypic behaviors in order to simulate other actors' behavior. They imagine their own behavior by simulating their own decisional mechanisms and imagine the behavior of the other actors using prototypic FCMs. They can use their imaginary worlds to choose one strategy amongst several possibilities, not through logical reasoning but rather by behavioral simulation. Thus, they will be able to predict evolutions within the environment.

4.3.3 Learning

In this section, we present a method for adapting prototypic behavior through imitation in real-time. Agents observe their environment (*i.e.* other agents), which allows them to simulate the behavior of other entities in their imaginary worlds with prototypic FCMs. The idea is to provide a more relevant simulation by adapting prototypic FCMs through imitation. The modification of prototypic FCMs reduces the difference between predictions in the imaginary worlds and reality [Schultz et al., 1996a]. We assumed that agents have sensors to deduce information relating to prototypic FCMs. This means estimating sensor and effector values that will fuzzify sensor values, and comparing the result of defuzzified motor concept activations with the model's effector values.

The learning mechanism consists in retrieving the simulation results in the imaginary world, comparing them to what happened in the virtual world, and thereby adapting the prototypic FCM. To be consistent with the knowledge-based modeling of the behaviors, leading the designer to explicit both concept and links of the FCM, the learning solely consists in adapting the weights of the causal links between concepts of the prototypic FCM. Therefore, the learning algorithm does not modify either the structure of the FCM's influence graph, or the fuzzification of the sensors, or the defuzzification of the motor concepts. This modification in the causal connections between concepts could be controlled by the expert. In particular, the expert could verify the FCM's structure, impose signs for some links and define some link interval values. This is what we call "meta-knowledge about learning".

4.3.4 Why not modify the FCM structure?

FCMs have the ability to visually represent behavioral expertise by means of a semantic graph. The concepts, the causal links between them, and these links' signs are assigned semantic descriptions. In this case, learning does not alter the structure of the influence graphs, so that the behavioral coherence as seen by a human observer may be preserved [Papageorgiou et al., 2004,Papageorgiou and Groumpos, 2005,Papageorgiou et al., 2006]. Nor does it alter the fuzzification of the sensors or the defuzzification of the motor concepts which remain unique for each agent.

4.3.5 Algorithm

Kosko [Kosko, 1988] proposed two different Hebbian learning methods [Hebb, 1949]. One is based on the correlations between activations [Kosko, 1992] and the other on a correlation of their variations (differential Hebbian learning) [Dickerson and Kosko, 1994]. Differential learning modifies only the links associated with correlated variations in the concepts' activations, while non-differential correlation learning runs the risk of inappropriately modifying all the links. Kosko's differential learning is based on the knowledge of a limit cycle which takes all concepts into consideration, and which is provided by an expert. However, we cannot use such a limit cycle because only estimated model sensors and effectors can be observed, and the FCM which generated them is unavailable. In addition, Kosko's differential learning makes the assumption that external activations are constant, however, the virtual world is a dynamic system and external activations evolve over time. It is therefore necessary to adapt Kosko's differential Hebbian learning to simulate realistic behaviors in a dynamic virtual environment.

The adaptation algorithm that we propose is a four-stage iterative cycle (see Figure 4.7):

1. *Model estimation.*
   The agent estimates model-sensors and model-effectors through observation. We make the assumption that these features are available.

2. *Simulating prototypic behavior.*
   Sensors are fuzzified into external perceptive concept activations. The FCM's dynam-
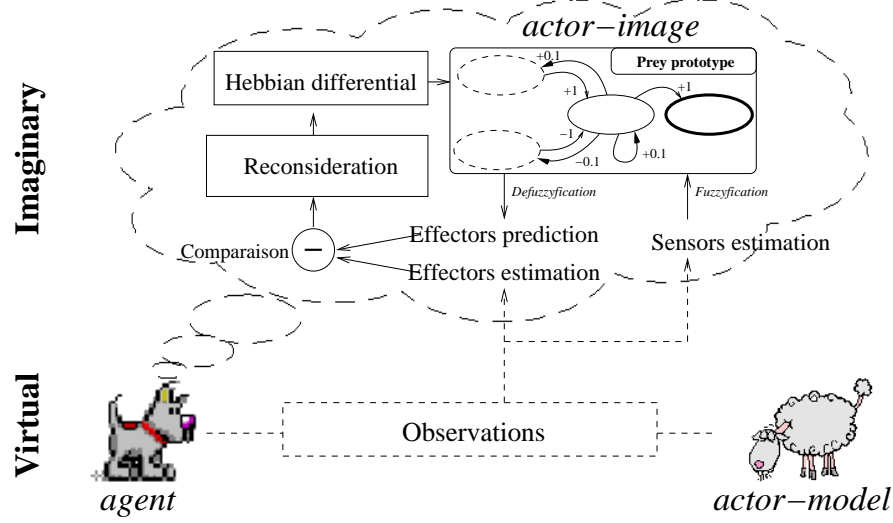
The "dog" agent and the "sheep" actor-model evolve within the virtual world. The dog possesses is own imaginary world in which it can simulate prototypic behavior from a library of behaviors, containing the "prey". In its imaginary world, the dog tries to imitate the sheep by an actor-image using the prey prototype. The imitation is conducted in real-time *according to the events occurring in the virtual world, while comparing the observed effectors of the sheep in the virtual world with the predicted effectors by the prey prototype in the imaginary simulated world (as estimated by the sheep's sensors). If necessary, discrete reconsideration takes place at the level of the prey's internal activations in order to reduce this difference. The prey prototype is then updated via differential Hebbian learning.*

FIGURE 4.7 – The agent, the actor-model and the actor-image

ics are calculated, and image-effectors are obtained, by activating inner motor-concept defuzzification.

3. *Calculating reconsiderations.*
Image-effectors and model-effectors are compared, generating a set of desired pseudo-activations. These pseudo-activations are obtained by going back up along the graph from motor concepts towards perceptive concepts, without modifying links and by using meta-knowledge about learning.

4. *Updating causal links.*
FCM causal links are updated by applying discrete differential Hebbian learning to the sequence corresponding to the transition from FCM activations to desired pseudo-activations.

In the following sections we shall examine these four stages in more detail.

4.3.5 - A   Observation

During the first stage, the agent measures features of the actor-model which are required for model-sensor and model-effector estimations. For example, the agent "dog" estimates the

distance between the sheep and a predator by the difference in position at that instant t, and it estimates sheep's speed by the difference in the sheep's positions at the instants t and $t-1$.

4.3.5 - B   Prediction

The second stage simply corresponds to the classic use of FCMs for controlling a virtual actor and determines image-actor FCM activations at $t + \delta t \approx t$ in the imaginary world, according to model-sensor estimation and FCM dynamics with N iterations:

$$a\big(t+\tfrac{I}{N}\delta t\big) = S\left(G(f(t), L^T \cdot a\big(t+\tfrac{I-1}{N}\delta t\big))\right)$$
$$\text{for } I = 1, \cdots, N \ ; \quad \delta t \ll 1 \tag{4.2}$$

N equals the length of the longest acyclic path added to the length of the longest cycle in the influence graph in order to make sure that sensor information is spread to all nodes. n is the FCM concept number, $f=(f_i)_{i\llbracket 1,n\rrbracket}$ external activations coming from sensor fuzzification, $a=(a_i)_{i\llbracket 1,n\rrbracket}$ internal activations, $L=(L_{ij})_{(i,j)\llbracket 1,n\rrbracket^2}$ link matrix, $G : (\mathbb{R}^2)^n \to \mathbb{R}^n$ a comparison operator and S a standardization function transforming each coordinate by the sigmoid function: $\sigma_{(\delta,a_0,\rho)}(x)=\frac{1+\delta}{1+e^{-\rho(x-a_0)}}-\delta$, with parameters $(\delta,\rho,a_0)\in\{0,1\}\times\mathbb{R}_*^+\times\mathbb{R}$. FCM motor concept defuzzification at $t + \delta t \approx t$ provides image-effectors. To clarify, we note a the resulting inner activations $a(t + \delta t)$ in the following paragraphs.

4.3.5 - C   Reconsideration

The third stage recursively generates sets of pseudo-activations $(P_i)_{i\in\llbracket 1,n\rrbracket}$ representing the orientation for FCM dynamics. This is done by moving back up the influence graph from motor concepts towards perceptive concepts, proposing pseudo-activation values according to meta-knowledge about learning, and bringing image-effectors closer to the estimates of model-effectors. We did not use the gradient backpropagation method [Rumelhart and Mcclelland, 1986] because FCMs are cyclical processes and their topology is not organized in layers (recurrent links). Furthermore, the gradient backpropagation method does not hold graph semantics and we wanted to be able to apply specific meta-knowledge to specific nodes. We shall now look more closely at the recursive process.

Initialisation $m = 0$. Entering into the FCM from effectors. A set I_0 represents indices of concepts defuzzified onto image-effectors. For each $i \in I_0$, we apply learning meta-knowledge. Two potential pseudo-activations $p_i^{\pm} = \sigma(a_0 \pm \frac{2\alpha_i}{\rho})$ simulate an active/inactive concept C_i, $\alpha_i \geq 1$ representing choice radicality. Including the a_i value, there are three possible pseudo-activations $p_i = a_i$, p_i^+ or p_i^- for each C_i. The $3^{\text{Card}I_0}$ combinations are defuzzified and compared to model-effector estimates. The best combination $(p_i^{0,\{\}})_{i\in I_0}$ is retained (the 0 deals with defuzzification and the {} is a set of future labels). $\forall i \in I_0$, $P_i = \{p_i^{0,\{\}}\}$. The other pseudo-activation sets $(P_i)_{i\in(\llbracket 1,n\rrbracket \setminus I_0)}$ are empty. We propose this discrete reconsideration rather than a gradient-scaled calculation [Mozer, 1995, William and Zipser, 1995]. A discrete choice like this facilitates the agent's decision-making process.

Progression from m to $m+1$. Let $I_m \subset \llbracket 1,n\rrbracket$ be the index set of concepts whose desired pseudo-activation set is not empty. For $i \in I_m$, note a_i (respectively f_i) internal (respectively

external) activation of concept C_i, $P_i = \{p_i^{k_1,\{\cdots\}}, \cdots, p_i^{k_L,\{\cdots\}}\}$ its desired pseudo-activation set whose cardinal equals L and $J \subset [\![1, n]\!]$ the index set of concepts which are causes for the concept C_i (*i.e.*:$L_{ji} \neq 0$) and such that the edge from C_j to C_i has not been studied: $\forall \lambda \in [\![1, L]\!]$, $j \neq k_\lambda$. We will calculate pseudo-activations P_j for $j \in J$ as follows:

▷ For each $j \in J$, we apply learning meta-knowledge: two potential pseudo-activations, p_j^+ and p_j^-, are calculated using the formula (4.3) so that their influence on a_i causes a clear choice between an active C_i or an inactive C_i. This accounts for external activations, with $\alpha \geq 1$ representing the choice radicality:

$$p_j^{\pm} = \left(a_0 \pm \frac{2\alpha_j}{\rho} - f_i - \sum_{l \neq j} L_{li} a_l \right) / L_{ji} \qquad (4.3)$$

Note that a_0 and ρ are FCM sigmoid function parameters (see fig 4.2); α is a learning algorithm parameter.

▷ Then we randomly select a $\lambda \in [\![1, L]\!]$. This gives $p_i^{k_\lambda,\{\cdots\}} \in P_i$ and we choose between the possible $3^{\mathrm{Card}J}$ combinations $p_j^i = a_j$, p_j^+ or p_j^- for $j \in J$, the one $p_j^{i,\{\cdots,k_\lambda\}}$ which gives a C_i activation $\sigma\left(G_i(f_i, \sum_j L_{ji} p_j^i) \right)$ the nearest to $p_i^{k_\lambda,\{\cdots\}}$,

▷ Thus we obtain a new set of concept indices with a pseudo-activation set which is not empty: $I_{m+1} = I_m \cup J$ with $P_j = P_j \cup \{p_j^{i,\{\cdots,k_\lambda\}}\}$ for $j \in J$.

Termination. If for each $i \in I_m$, the corresponding J set is empty, that means every edge belonging to the paths arriving in $(C_i)_{i \in I_0}$ has been studied.

We use a discrete method by proposing three pseudo-activations. The discrete method chosen will allow us on one hand to limit the calculations and on the other, to represent a radical choice. Our argument is that to learn, the proposed modifications need to correspond to radical choices rather than minor alterations.

4.3.5 - D   Update

The fourth and final stage modifies the matrix of the FCM's links in such a way that its dynamics are directed towards resulting behavior similar to that of the actor-model. We use discrete Hebbian learning to pass from internal activations a to questionings p calculated during the previous stage. Unlike Kosko who used a limit cycle and a learning rate decreasing towards zero over time to ensure convergence (see [Dickerson and Kosko, 1994] page 186), we only make them learn the passing from internal activations a to reconsideration p in order to modify the links without creating cycles while maintaining a constant learning rate $r(t) = R$. A cycle would teach not only the passing from a to p, but also from p to a, which is inappropriate. Our learning rate is constant so that the agent will conserve its adaptive nature. Theoretically, there is nothing to prevent the learning rate from being modified over time. This can be achieved by making it follow a series decreasing towards zero, with an infinitely decreasing associated series, as, for example, $r(t > t_0) = \frac{R}{t - t_0}$. This would ensure

that the weights of the FCM's edges would converge, but adaptability would decrease over time.

Formally, noting $\mathcal{A} \subset [\![1,n]\!]^2$ the edge set of the FCM, $\beta \in]0; 1+\delta[$ a sensitivity level and $s : \mathbb{R} \rightarrow \{-1, 0, 1\}$ the discrete function $s(x) = -1,\ 0$ or 1 if $x \leq -\beta,\ -\beta < x < \beta$ or $x \geq \beta$ respectively, the learning algorithm corresponds to the following equations:

$$
\begin{aligned}
&\forall (i,j) \in \mathcal{A},\ \text{if } \exists k \in [\![0,n]\!],\ p_j^{k,\{\cdots,i,\cdots\}} \in P_j, \\
&\text{with this } k\ : \\
&\begin{cases}
\Delta_i = s(p_i^{j,\{\cdots\}} - a_i),\ \Delta_j = s(p_j^{k,\{\cdots,i,\cdots\}} - a_j) \\[2mm]
L_{ij(t+1)} = \begin{vmatrix} L_{ij(t)} + R(\Delta_i \Delta_j - L_{ij(t)}) &,\ \text{if} & \Delta_i \neq 0 \\ L_{ij(t)} &,\ \text{if} & \Delta_i = 0 \end{vmatrix}
\end{cases} \\
&\text{otherwise } \mathcal{A}_{ij} \notin \{\text{path to effectors}\} :\ L_{ij(t+1)} = L_{ij(t)}
\end{aligned}
\tag{4.4}
$$

It must be noted that we preserve coherence in the modification of links as specified in the initial prototype provided by the expert. Link emergence, link suppression, or modification of a link's sign are therefore forbidden. Furthermore, some links can be maintained in terminals $\mathcal{B}_{ij} = [L_{ij}^{min}, L_{ij}^{max}]$ so that the modified behavior might remain consistent with the expert's initial description: if $L_{ij(t+1)} < L_{ij}^{min}$ then $L_{ij(t+1)} = L_{ij}^{min}$ and if $L_{ij(t+1)} > L_{ij}^{max}$ then $L_{ij(t+1)} = L_{ij}^{max}$.

4.3.5 - E Complexity

The complexity of this algorithm is a polynomial function of the number n of FCM concepts, and even $\mathcal{O}(n)$. For an expert, a concept's causes are always very limited in number (seldom more than seven). Therefore, the number of edges arriving on each concept is increased by M ($M \approx 7$). $\mathrm{Card}J \leq M$. $3^{\mathrm{Card}J}$ is thus increased in practice, whatever the number of concepts involved in the FCM. The same applies to the calculation of FCM dynamics with a complexity of $\mathcal{O}(n)$ whereas they could seem to be $\mathcal{O}(n^2)$, due to the great number of zeros in the link matrix. The number of not null links in a column is no more than M, whatever n might be. This algorithm can thus be implemented for use in real-time.

4.3.6 Application

Based on the sheepdog environment described above, we implemented three applications. First, the dog learns one way of rounding up sheep by imitating a human operator or another dog. In these cases, the prototypic FCM used is the dog's own FCM. Second, the dog's prey prototype adapts to a given sheep in real time. This application is described in this section. Third, fearful sheep learn how to be surrounded by other sheep. The sheep remain frightened but no longer flee when they come upon a dog. Immobilizing fearful links means that the sheep's behavior can be adapted whilst at the same time preserving a fearful "personality".

To simulate sheep behavior, the dog uses prototypic FCMs of prey from its behavioral library. The dog actually represents each sheep's behavior through prototypic "prey" FCMs

in its imaginary world, with each sheep being associated with its own prototype. The dog can therefore simulate the sheep's behavior in order to make predictions and to test different strategies.

The prototypes will adapt to each sheep through imitation. One FCM controls the prototype's speed and another controls its direction. Comparisons between the result of the imaginary and the virtual worlds are used to adapt prototypic FCMs in real-time through learning. Figure 4.8 illustrates the modification through imitation of a prototype's speed and the representation of one sheep's speed using the imaginary world. We chose this set of learning periods to ensure that the process would converge.
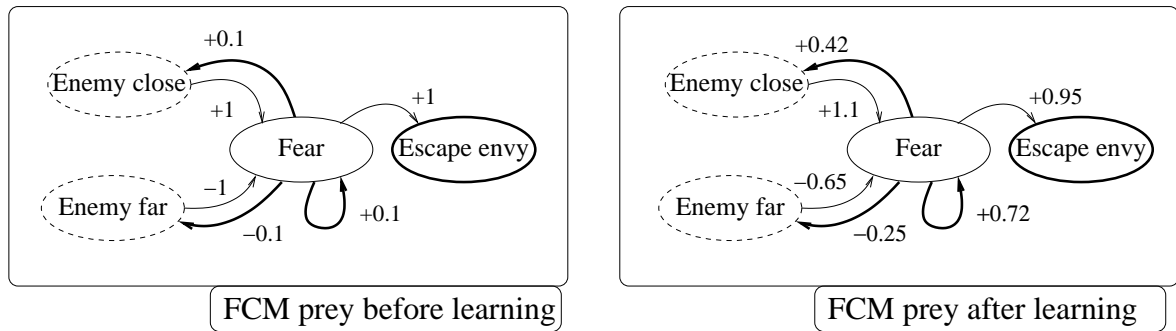
FIGURE 4.8 – An FCM of perceptive prey from the library of prototypic FCMs which adapt themselves by learning.

In order to imitate, the dog first observes the sheep. It adapts the prototypic prey behavior allowing it to simulate the sheep's behavior in its imaginary world. By observing the sheep, it estimates the information necessary for the fuzzification of the prototype (*Phase 1 : observation*). Estimated sensor values are fuzzified in activating the concepts "Enemy close" and "Enemy far". The prototype dynamic occurs and, by defuzzifying the activation of the motor effector "Escape envy", we obtain the image effector (*Phase 2 : prediction*). This corresponds to the dog's representation of the prey's speed. The image effector from the prototype is compared to an estimation of the sheep's effectors and this comparison is used to calculate a set of pseudo-activations associating the desired modifications to the FCM's links (*Phase 3 : reconsideration*).

In Figure 4.9, we compared the simulation of the sheep's behavior obtained from the prototype in the imaginary world ("Prey image"), with the sheep's behavior in the virtual world ("Sheep Model"), both before and after learning, while the dog performs the same trajectory ("Dog"). The modelled sheep is controlled by the map in Figure 4.3b (with $\lambda = 0.5$ and $\gamma = 0.6$).

The human operator decides on the training period from start to finish. The dog's acquired imitation experience illustrated in Figure 4.9 represents around 100 cycles, during which the dog approaches and moves away from the sheep twice. If learning were to become permanent, parameter λ in the "prey" prototype would be reduced to as little as 0.3 when the sheep remains at a distance from the dog (over 1,000 cycles) but which quickly (below 10 cycles) goes back to a value of around 0.7 as soon as the dog begins to move towards the sheep. This proves the need for a constant learning rate: adaptability remains extremely responsive no matter what the duration of the learning.

We then went on to generate a herd of 30 sheep with different "personalities" (differences λ and γ in the sheep's FCM). We assigned the dog a "prey" prototype for each sheep and requested the same number of parallel learning processes as there were sheep in the herd. We obtained significant predictions, with each prey prototype adapting quickly to each sheep (relative stability of the coefficient after 1,500 cycles; the time required for the dog to approach each sheep at least once). However, a simultaneous learning technique would not be possible for larger herds, as when there are more than 300 sheep, the dog no longer has the time to learn in real-time [3].
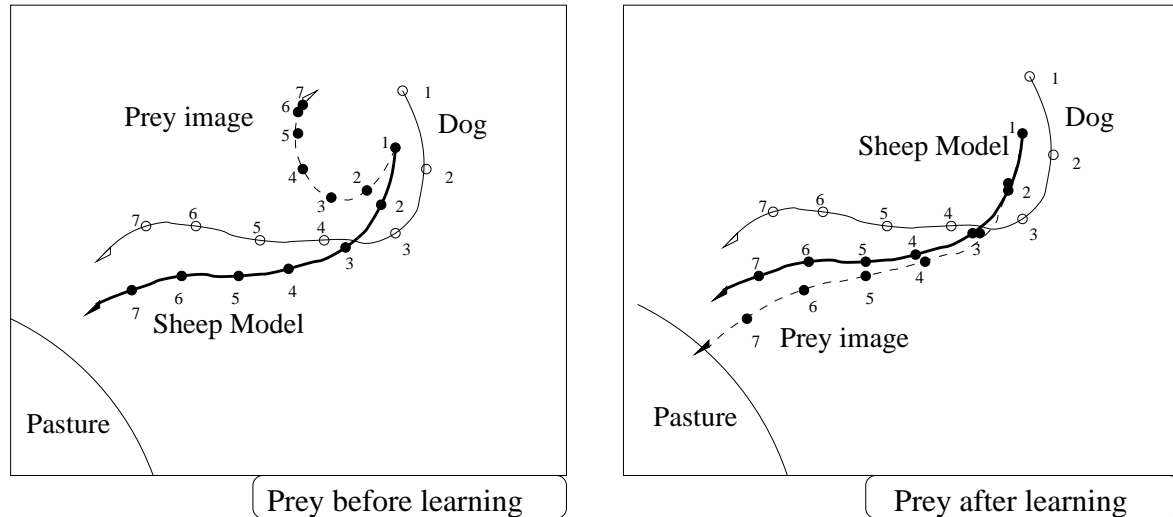
FIGURE 4.9 – More pertinent predictions can be obtained from the imaginary world by using imitation learning.

4.4 Conclusion

In order to be believable, autonomous behaviors must be driven, not only according to external stimuli, but also according to internal emotions such as fear, satisfaction, love or hate. We described these behaviors using fuzzy cognitive maps (FCMs) where these internal states are explicitly represented. Instead of the conventional use [Dickerson and Kosko, 1994], we **embedded FCMs** into each agent. This defines the decision-making period of their lifecycle. The agents implemented with FCMs are not only sensitive, but perceptive; their behavior depends on their internals states retroacting on the sensors.

We described the use of FCMs as a tool for modeling the behavior of virtual actors improvising in free interaction. We highlight specific modeling features which can prove particularly advantageous, such as the flexibility concerning system design and control, the comprehensive structure and operation, adaptability to problem-specific prerequisites, and the capability for abstractive representation and fuzzy reasoning.

---

[3] Our models were implemented using oRis [Chevaillier et al., 2000] language, and were made on a basic linux PC with a 400MHz CPU and 128MB of RAM.

Our agents possess a behavioral library made up of prototypic FCMs. While acting in the virtual world, the prototypic FCMs allow the agent to simulate the behavior of other actors in its imaginary world. These FCMs simulate different strategies, allowing the agent to make predictions. We use FCMs to predict behavior, not by formal reasoning as it was realized for the reasoning on beliefs, the distributed decision and the organization of agents in interaction from the global standpoint [Chaib-Draa, 2002], conceptual graphs for human experts, but by **behavioral simulation**.

We presented a learning algorithm allowing the prototypic FCM to adapt through observation. Our algorithm changes the weights of FCM connections. It does not, however, modify the structure, the fuzzification of the sensors, or the defuzzification of the motor concepts. The applications depict a sheepdog using a prototypic FCM to predict the behavior of a herd of sheep.

The following points are the major limits of our proposal. Currently, the **prototype choice** (for internal simulations and for learning) is provided by the designer of the simulation. Moreover the **learning periods** are not chosen by the agent, they are imposed by the designer.

Transferring these competencies to the level of agents will increase their autonomy, while automating the entire process. Consequently, future research will aim to implement a process that selects a prototype in the library through observation of the model behavior to be imitated. Furthermore, the learning period will be selected automatically. We are also working on adapting the fuzzy transformations associated with fuzzification and defuzzification.

Contents of Chapter 5

Chapter 5

GEMEAU — a generic model for a family of classifiers systems

Classifiers systems are tools adapted to learning interactions between autonomous agents and their environments. However, there are many kinds of classifiers systems which differ in a number of subtle technical ways. There is no one rule for choosing among them according to a given problem. This chapter analyzes the major kinds of classifiers systems in order to suggest a generic model (called GEMEAU) that is common to all of them. GEMEAU was developed for a number of different applications and it is applied to timetables problem in order to easily test differents hypotheses.

5.1   Introduction

DEFINING the behavior of autonomous artificial entities is faced with the problem of acheiving a model able to account for the link between perceptions and actions in an artificial and efficient manner. There are a great number of proposed solution to this issue which require detailed description which are difficult to achieve, either because they require a definition based on *a priori* rules and symbols [Carver and Lesser, 1992, Mateas, 1999, Cavazza et al., 2001], or because they are subject to configuration difficulties and behavioral indeterminism [Brooks, 1990, Maes, 1989]. Another solution would be to define the entities' initial, approximate behaviour, which would then adapt according to its environment. This solution is implemented by classifiers system and offers the advantage of being made up of an ensemble of competing rules and incorporating learning processes by choosing and improving these rules. A great deal of literature exists on the subject [Cliff and Ross, 1995, Stolzmann, 1998, Lanzi and Wilson, 1999, Tomlinson and Bull, 1999b, Sigaud and Wilson, 2007b]. A number of authors have put forward different variations of the approach, each offering different mechanisms adapted to specific problems. Our objective is to be able to test and advance these mechanisms without difficulty, which is why we are interested in designing and implementing a generic model.

This chapter is organized in the following manner: firstly, we present the general mechanisms of classifiers system (section 5.2) and demonstrate their use in ZCS and XCS systems (section 5.3). We then go on to present a generic model, called GEMEAU, which integrates these mechanisms, and with which we can easily test different versions (section 5.4). Next we explain how we applied this model to different types of applications: multiplexers, *Woods* environments (section 5.4) and the optimization of a distributed resolution algorithm able to adapt to the problem of generating timetables (section 5.5). Finally, we look to the possible future perspectives concerning the use of classifiers system in adapting educational strategies for virtual environments for training (section 5.6).

5.2 Classifiers systems

5.2.1 Principles

Classifiers system manage a combination of rules referred to as "condition-action" classifiers, which can be counterbalanced by quality attributes to obtain a learning system. The system perceives its environment, deduces the applicable rules, carries out an action as a result of these rules. The system is able to receive a credit from the environment which it then uses to modify the rules or their quality attributes. It is the quality of a rule, associated with the correspondence between its conditioning and its perception of the environment, which determines its choice. Through experimentation, classifiers system can therefore be used to learn the association between conditions and actions, thus maximizing credit intake. In order to avoid a combinatorial explosion of the quantity of rules, they are generalized; they apply to different perceptions of the environment. Mechanisms which allow the creation, enrichment (specialization/generalization), or destruction of these rules must therefore be used. Evolutionary algorithms are often used to do this, even though other heuristic approaches are available. The qualities of the rules are modified dynamically through reinforcement learning, and the rules themselves are modified by genetic algorithms.

5.2.2 Formalization

In this section we propose the incremental and generic formalization of classifiers systems, and gradually introduce learning mechanisms.

5.2.2 - A Basic Structure

The global structure of a classifier system, is a 7-uplet (`Ii`, `[P]`, `[M]`, `[A]`, `Matching`, `Selection`, `Io`) :

▷ `Ii` is the interface input due to which each `Perception` within the environment corresponds to a binary code.

▷ **[P]**, (*population*), is the ensemble of the system's classifiers, coded by a succession of n bits[1]. The generalizing representations contain # symbols which correspond to an indeterminate value. A rule is a (C, A) pair with $C \cup A \in \{0, 1, \#\}^n$ where :

- C : the condition for application of the rule.
- A : the action(s) associated with the application of the rule.

Let us take the example of a robot with four 'all-or-nothing' sensors and one action. The input interface converts the state of the sensors into a binary value and the output interface triggers the action depending on the action's bit value. Thus, a $\{011\#, 1\}$ rule means that the rule is applicable if the first sensor is inactive and the two following sensors active. The state of the fourth sensor has no influence, and applying the rule triggers the action.

▷ **[M]** $\subseteq$ **[P]** is the set of classifiers of which the condition element pairs with the perceived environmental information during a selection cycle. This is known as *Match-set*.

▷ **[A]** $\subseteq$ **[M]** is the set of classifiers representing the selected action. This is known as *Action-set*.

▷ **Matching** is the mechanism which makes the transition from **[P]** to **[M]** possible. This is generally achieved using a matching rule between C and the information derived from **Ii**. This rule is able to interpret the generalization symbols that make up the classifier conditions.

▷ **Selection** is the mechanism which makes the transition from **[M]** to **[A]** possible. Depending on the details of the different versions of classifiers systems, it is able to determine the desired action.

▷ **Io** is the output interface through which the activated **Action** corresponds to a binary code.

5.2.2 - B   Learning

Learning occurs due to an evaluation of the quality of the rules represented by one or a number of additional parameters. The definition of a classifier is thus extended to a $R = (C, A, f)$ triplet where f characterizes its quality. Learning developed by **Rewarding** the rules, by altering their quality using reinforcement learning algorithms and by **Generating** rules using evolutionary and heuristic covering algorithms. The dynamics of learning classifiers systems are therefore based on the following cycle: **Perception** / **Matching** / **Generation** *(covering)* / **Selection** / **Action** / **Reward** / **Generation** *(evolutionary algorithm)*. The pseudocode for the cycle is presented in figure 5.1.

5.2.2 - C   Selection Mechanism

Selection is guided by the quality of the rules, which are grouped together depending on their **[A]** element. Often, a 'wheel of fortune' mechanism[2] is applied, which means that each package

---

[1] Even if certain systems work with other alphabets [Matteucci, 1999, Wilson, 2000, Heguy et al., 2002].

```
t := 0       // start with an initial time
// and randomly generated population of classifiers
initClassifierPopulation( P(t) )
// test for cycle termination criterion (time, fitness, etc.)
while not done do
    t := t +1
    // Perception : 1. detectors check
    Ii(t) := readDetectors(t)
    // Matching : 2a. compare [P] to Ii and save match set in [M]
    M(t) := matchClassifiers( Ii(t),P(t),Matching )
    // Generation (1) Covering
    // : 2b. create rules (matched with Ii (if M empty or criterion1)
    if ( M.size < criterion0 || criterion1 ) then
            M(t) := cover( Ii(t),P(t),Covering )
    // Selection : 3. rule selector in [A]
    A(t) := selectClassifiers( M(t),Selection )
    // Actions : 4. new messages through Io
    Io(t) := sendEffectors( A(t) )
    // Credit assignment (1) : 5. receive payoff from environment
    r := receivePayoff(t)
    // Credit assignment (2) : 6. distribute payoff/credit to classifiers
    P(t) := distributeCredit( r,P(t),P(t-1),Payoff )
    // Generation (2) A.E. : 7. EA (usually a GA) is applied to [P]
    if ( criterion2 ) then
            P(t+1) := reviseRules(P(t),Evolutionary_Algo)
```

FIGURE 5.1 – Pseudocode representing the seven stages of a learning classifier system's cycle

has a probability proportional to its capacity to be selected. There are different versions of selection algorithms which favor package size, the quality of the best rule, a compromise between the two, etc.

5.2.2 - D   Credit assignment Mechanism

The credit assignment mechanism distributes credit to the rules that have contributed to its acquisition. It increases the quality of the rules triggered prior to the acquisition of the credit and decreases that of the others. Its definition affects the relevant length of a series of actions: i.e. the number of rules in a sequence considered necessary in order to achieve a certain goal.

5.2.2 - E   Generation Mechanism

The generation mechanism must both minimize the number of rules and conserve those which assist in achieving credit. A good rule is therefore a high-quality generalizing rule (relative to the others). The two generation (*rules discovery*) mechanisms used are *covering* and *genetic algorithms*.

▷ *Covering* enables the creation of rules when no classifiers match the perception of the environment. This means that the population [P] has an insufficient number of rules

---

[2] The wheel of fortune mechanism consists of picking elements randomly, so that their probability of being chosen is proportional to their selectivity.

enabling it to put forward an action relative to the situation. *Covering* can also create rules when the qualities of matched classifiers are considered insufficient. In both cases, new rules are created with more or less general suitable condition elements. In this case, the probability of obtaining a $\#$ is a parameter which must be fixed. The action element is chosen randomly and the *Quality f* is often the mean of the [P] qualities. Using this method, the system can be initialized with an empty population [P] and therefore avoid the generation of rules which do not correspond to any possible state in the environment.

▷ *Genetic algorithms* [Holland, 1975b,Goldberg, 1989] are used to generate new rules from pre-existing ones. Genetic algorithms can intervene in rules, placing themselves in [P] or in [A], depending on the versions of classifiers systems [Wilson, 1994,Wilson, 1995]. This mechanism enables the systems to adapt more quickly to dynamic environments. The frequency of use of genetic algorithms is an important factor with regards to the cycle of a classifier system, which can noticeably influence its learning performance. There are two main types of classifier system: *Michigan* [Smith, 1980b] and *Pittsburgh* [Holland and Reitman, 1978]. The *Michigan* approach applies the genetic algorithms by using only one system where each rule is unique. The *Pittsburgh* method considers an individual as a whole population (i.e. a set of rules).

5.3 Different versions

The complexity of the early systems such as CS1 yielded no conclusive results[3]. Indeed, CS1 can send itself messages using the message list and thus create cycles of internal inferences which tend to develop and maintain parasitic rules. Unlike certain experiments which aim to correct the performance deficiencies of the original systems, Wilson instead chooses an approach which consists of simplifying them, and thus proposes ZCS. By eliminating the message list, he reduced the original systems down to their essential elements whilst at the same time conserving their original architecture.

5.3.1 ZCS

The Zeroth level Classifier System (ZCS) was presented by Wilson as a *Michigan*-type system of classifiers [Wilson, 1994]. This type of classifier system uses rules in the traditional $R = (C, A, f)$ triplet form. It specifies the following mechanisms:

Rule `Selection` takes into account the *quality* of [M] classifiers. During use, it is generally subject to the 'wheel of fortune' mechanism.

`Credit` is subject to a mechanism much like Q-Learning [Sutton, 1984,Sutton, 1988c] : the *Bucket Brigade* [Dorigo and Bersini, 1994]. The environment's credit is the source of

---

[3] CS1 emerged in 1978 [Holland and Reitman, 1978]: a situation or a "perceived state" is stored in a limited perspective memory referred to as a message list. This notion is assimilated within the concept of "short-term memory" found in cognitive sciences.

the credit chain which passes through each of the classifiers that participated in trigger-ing the action, and ends with the classifiers at the origin of all action. Each rule takes its corresponding section and gives what is left to the other classifiers via a *discount factor*. Algorithms [Holland, 1985] introduce an economical system to the classifier population `[P]`. The idea is to assimilate classifiers and their environments to financial officers. The qual-ity of a rule can therefore be considered to be the agent's capital. The market mechanism implements competition between the agents concerned by introducing the environment in order to be able to participate in bids. For each `[P]` classifier, three conditions are intro-duced: a *Bid Cbid*, a *Tax Ctax* and a *Credit Cr*. Let us consider a classifier of the form: $R = (C, A, f, s, Cbid, Ctax, Cr)$. The *Quality f* of each `[P]` classifier is updated at each life cycle, according to the formula: $f(t) = f(t-1) - Cbid * f(t-1) - Ctax * f(t-1) + Cr$.

`Generation` is conducted with the help of *covering* or of genetic algorithms applied to the population `[P]`, when considering the quality of classifiers as a selective value. `Generation` acts at a constant frequency. The mutation operator applies "parents" from the 'wheel of fortune' method on `[P]`. The new classifiers replace the weakest rules in order to maintain a constant population. This localized deletion selects rules using the 'wheel of fortune' method on `[P]`, by considering the inverse value of the quality. The quality of the new rules is initialized by the mean value of the qualities of the "parents". In the case of *covering*, it is initialized by the mean values of the qualities of `[P]`.

Different parameters are required in order to use this system [Wilson, 1994]. We noted population size, the classifiers' initial quality, the rate of bids, the rate of rewards, the *covering* rate, the frequency with which genetic algorithm are called upon, crossing rate, and the rate of mutation.

On a superficial level, we can show how an entity's ability to perceive its environment can be broken down according to two models:

1. information extracted from the entity's immediate perception provides all of the nec-essary information for choosing the best action in any situation. In this case, the environment is *Markovian*. The entity does not need to memorize the system's history in order to master the environment at a given moment; its immediate perception alone is sufficient.

2. information extracted from the entity's immediate perception provides partial infor-mation concerning the environment. In this case, different situations may exist which appear identical to the entity, but which require different optimal actions. Consequently, the entity cannot choose the most appropriate action by considering its immediate sen-sory information alone. The environment is therefore *non-Markovian*. In order to be able to differentiate between these states, the entity must take its background into account and can use explicit (internal register) or implicit (links between decisions) memory.

Some systems suggest extending ZCS capacities to non-Markovian environments:

▷ ZCSM (Memory) [Cliff and Ross, 1995] adds temporary memory to ZCS. Each of the sections (condition and action) of the rules is extended with a sub-chain of internal bits: memory. The choice of rules therefore depends on the value of the memory which is also

modified by the rules. Using this method, the capacities of the ZCS can be increased to non-Markovian environments, the difficulty being to define an appropriate size for this memory.

▷ ZCCS (Corporation) [Tomlinson and Bull, 1999b] defines the links between the rules. Each rule is linked to both the rules that precede it and those that follow it. A corporation is an set of interlinked classifiers. The crossing method uses selected corporation rules.

5.3.2   XCS

XCS [Wilson, 1995] is a version of a classifier system based on ZCS. The specifications of such a system are as follows:

The rules are made up of three parameters which come together to complete the condition and action sections. The *Quality f* of the ZCS is broken down in the following manner:

▷ The *Payment prediction p* represents the reward expected by conducting the action A in the situations matched with C,

▷ The *Prediction error e* represents the deviation between the *Payment prediction p* and the true payment,

▷ *Fitness fit* is a function inverse to e; it represents the accuracy of the *Payment prediction p*.

Classifiers can be defined by $R = (C, A, p, e, fit)$ with C and $A \in \{0, 1, \#\}^n$ and $p, e, fit \in \Re$.

Selection is not based on the *Quality f* of the ZCS classifier but rather on the accuracy of *Payment prediction p*. Using this method we are able to select the classifiers that not only have optimal actions, but rather those which can also accurately relay the quality of the proposed action, whatever its position in an action chain. A calculation of predictions (*Prediction Array* PA) is required. For each action a_i present in [M], a prediction $p(a_i)$ is calculated. [Wilson, 1996] suggests a clear separation between the strategies of exploration and exploitation:

▷ For exploitation, selection is determinist and based on the highest prediction; genetic algorithms are inhibited.

▷ For exploration, selection is subject to probability; either the highest prediction is selected or selection is conducted at random.

Credit is not a *Bucket Brigade* [Dorigo and Bersini, 1994]. It is rather subject to a mechanism of "Q-Learning"; *Payment prediction p* represents the utility function $Q(s, a)$ of Q-Learning, and backpropagation of the reward uses Bellman's equation. Once the **Selection** is complete, the set of preceding actions [A]$_{-1}$ (the ensemble of [A] during the most recent

cycle) is modified by using a combination of the environment's last credit and the current maximal prediction `PA`. The system no longer searches for classifiers adapted to the problem, but rather searches for an approximation of the utility function $Q(s, a)$, without limiting itself to the classifiers that propose optimal action.

`Generation` is completed using *covering*. It is used to initialize `[P]` with an empty or very small-scale set. The process of genetic algorithms is no longer situated at the `[P]` population level, but instead at the `[M]` level, and is based on prediction error. *Fitness fit* is the selective value. If the error is great then the selective value is weak and vice-versa. Details of these operations can be found in [Butz and Wilson, 2002]. [Butz and Pelikan, 2001] showed that this method favors the containment in `[P]` of the most general classifiers. The classifier is retained as long as its error is low.

5.3.3   Supplementary systems

There are different versions of classifiers systems which aim to improve the model and/or the scope of its applications. We here distinguish between *anticipatory* and *hierarchical* systems.

5.3.3 - A   Anticipation

Anticipatory systems are based on the hypothesis that it is critical to anticipate the consequences that an action might have in order to define a behavior. The main systems are outlined below:

▷ The ACS (Anticipatory Classifier System) [Stolzmann, 1998] adds an *effect* for each rule, anticipating the changes in the world which might occur as a result of the condition's action section. A classifier's quality is therefore calculated according to the quality of the anticipation of the *effects* the rule has on the the world. The system therefore creates a model of the world which in turn enables it to plan its actions. This system proposes the use of heuristics rather than genetic algorithms for constructing rules.

▷ The YACS (Yet Another Classifier System) [Gérard et al., 2002] uses the same formalism as the ACS with different heuristics improving the speed of latent learning.

▷ The MACS (Modular Anticipatory Classifier System) [Gérard et al., 2002] is a formalism for latent learning which introduces partial learning; its rules do not anticipate the values of all of the sensors at once. This system offers new generalization and performance possibilities.

5.3.3 - B   Hierarchy

Hierarchical systems suggest the use of multiple classifiers systems exchanging information. The two main models are outlined below:

▷ ALECSYS (A LEarning Classifier SYStem) [Dorigo, 1995] is a hierarchical structure by which a complex task can be broken down into a number of simple tasks. Classifiers system play the role of referee, choosing whether or not to activate other systems that are learning basic behaviors.

▷ The OCS (Organizational Classifier System) [Takadama et al., 1999] is made up of different agents, each of which has a classifiers system which must complete a communal task. The specialization mechanism is the communication between agents which enables the system's efficiency to be increased.

5.4 Gemeau

Classical classifiers systems (ZCS, XCS, ACS, Hierachichal ...) go some way to finding optimal solutions in Markovian or non-Markovian environments. Nevertheless, as Sanza notes [Sanza et al., 2001], the improvements and supplementary systems are suitable only for specific cases and none of the models are able to supply an overall solution for all of the problems (XCS is only effective if the credits are discrete and of a fixed quantity; ACS is only useful if each action leads to a modification in the perception of the world, etc.).

There are, therefore, a great number of classifiers systems [Urbanowicz and Moore, 2009]. Developing and testing a variety of such systems take time and is not easy. Using the structure and dynamics analysis conducted previously we were able to come up with a generic background for a whole family of classifiers systems. Our architecture claims to be generic, in the sense that it can be used to implement ZCS and XCS systems (ZCS, XCS, ZCSM, XCSM).

5.4.1 Architecture

The architecture is displayed in figure 5.2 as a UML classes diagram. The system is called Gemeau. It is based around two components:(*interface with the environment* and *system*)

Interface with the environment determines the interactions between the system and environment, both of which are common to different classifiers systems . In our model, the different interfaces are implemented using three categories: *CS_ II*, *CS_ IO* and *CS_ R* (respectively entry interface, output interface and credit). Communication between the interfaces and the environment takes place in the form of messages, enabling the classifier system to have an implementation in parallel to the environment.

The *system* defines the elements and the mechanisms of our classifier system in concrete terms. Let us consider the following elementsă:

▷ A classifier (*CS_ Classifier*) is made up of several parts: condition (*CS_ Condition*), action (*CS_ Action*) and configuration (*CS_ Parameter*);

▷ The ensembles [P],[M],[A] and [A]$_{-1}$ are lists of *CS_ ClassifierList*-type classifiers;

We put forward the following mechanisms:

▷ The `Matching` mechanism, with which the classifiers that match the information coming from the environment can be extracted. It is included in the *CS_ ClassifierList* by the *match()* method.

▷ The `Generation` mechanism by *covering*, which creates rules according to the content of [M] after `Matching`. It is included in the *CS_ ClassifierList* by the *cover()* method which can be configured (notably for the number of #).

▷ The general (*CS_ System*) method represents the workings of a given cycle (*step()* method) using the pseudocode from figure 5.1.

▷ The `Selection` mechanisms of the winning (*CS_ SelectorAlgo*) actions (which must be able to differ according to the desired learning).

▷ The `Credit` mechanism, (*CS_ AOCAlgo*), modifying the classifiers' configuration.

▷ The `Generation` genetic algoritm, (*CS_ GeneticAlgo*), where different operators must be specified, i.e. crossing or mutation.

5.4.2 Use

GEMEAU can be specialized in order to obtain a ZCS (figure 5.2). Through inheritance, we can define:

▷ The `Rules` (*ZCS_ Classifier* derived from *CS_ Classifier*) having a configuration force (*ZCS_ Power* derived from *CS_ Parameter*);

▷ The 'wheel of fortune' type `Selection` mechanism `ZCS_RouletteWheel` derived from *CS_ SelectorAlgo*);

▷ The *Bucket Brigade*-type `Credit` mechanism (*ZCS_ BucketBrigad* derived from *CS_ AOCAlgo*);

▷ The `Generation` genetic algorithm (*ZCS_ GeneticAlgo* derived from *ZCS_ GeneticAlgo*) notably specifying that the selective value is the rule's strength.

The rest of the system uses the pre-existing default mechanisms such as *covering*. We implemented the XCS classifier system using the same techniques. In order to do so, we must overdefine *CS_ Parameter*.
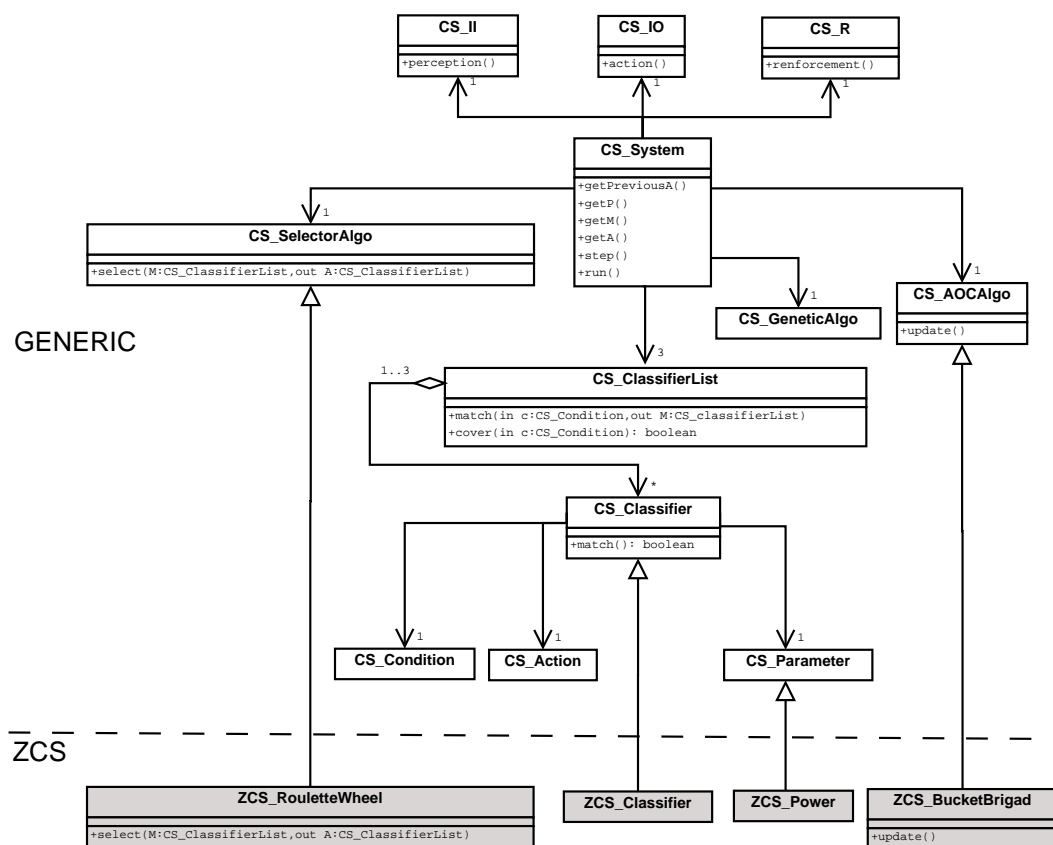
FIGURE 5.2 – UML diagram of GEMEAU augmented by a ZCS using inheritance

We can also easily add memory to ZCS in order to obtain a ZCSM. In that case, we must increase the result of perception using an internal classifier system register which is modified by part of the last action to have been carried out. Using GEMEAU, the *step()* method can be redefined simply by inheriting from the *CS_ System* (see figure 5.3). We implemented the XCSM classifier system using these same principles.

```
t := t +1
// Perception : 1a. code the perception
Ie(t) := readDetectors(t)
// Perception : 1b. add the internal register
Ie(t) := Ie(t) + Ir(t)
...
// Selection : 3. rules selector to [A]
A(t) := selectClassifiers( M(t),Selection )
...
// Memoire : 8. internal register
Ir(t+1) := extractPart( getActionPart( A(t) ) )
```

FIGURE 5.3 – Pseudocode representing the changes made to figure 5.1. An internal register is installed in order to add memory to a traditional system

By using these specialization and extension mechanisms we were able to use our architecture to implement and test ZCS and XCS family classifiers systems (ZCS, ZCSM, XCS, XCSM). Our perspectives are based on the implementation of supplementary traditional anticipatory or hierarchical systems (ACS: anticipation and ALECSYS: hierarchy [Dorigo, 1995]). Implementing these systems could well be less simple that for the family of ZCS and XCS, and the architecture may need to be modified.

5.4.3 Validation

One simple and frequently used evaluation environment is a multiplexer. Let us consider a multiplexer with an input of 6 bits $a_0, a_1, d_0, d_1, d_2, d_3$ and an *output* of one bit. a_0 and a_1 correspond to address bits. The multiplexer equation is $output = \overline{a_0}.\overline{a_1}.d_0 + \overline{a_0}.a_1.d_1 + a_0.\overline{a_1}.d_2 + a_0.a_1.d_3$. The output will be either the value of d_0, d_1, d_2 or d_3, depending on the address. The aim is to find this multiplexing function.

Using GEMEAU, we must simply determine the detectors and effectors that interface with the environment plus the reinforcement to be distributed, and then instantiate a *CS_ System* (see figure 5.4). The conditions and actions of the classifiers here correspond to the multiplexer's input and output respectively. The classifier system is rewarded when the rule selected corresponds to the multiplexing funtion.

Another advantageous evaluation environment is the *Woods* environment. It corresponds to a graphical representation based on an infinitely repeating pattern. It represents a forest and the aim of a situated agent is to find food. Within this forest there are insurmountable obstacles (trees) and areas of unrestricted movement. The perception of the agent corresponds to a representation of the 8 squares surrounding the occupied position. The simplest of these environments is *Woods1* (see figure 5.5a). This is a determinist Markovian environment[4] The

```
    /* Environnement */
        env = new EnvironmentMultiplexer(nbEntries,nbOut);
    /* Relation with the environment */
        detector = new CS_II_Boolean(env);
        effector = new CS_IO_Boolean(env);
        reinforcement = new CS_R(env);
    /* System */
        system = new CS_System();
        system->setDetector(detector);
        system->setEffector(effector);
        system->setReinforcement(reinforcement);
    /* Activity */
        system->run();
```

FIGURE 5.4 – Implementation of our generic architecture for a multiplexer-type environment

Woods100 (see figure 5.5b), however, is non-Markovian. Indeed the optimum displacement from square number 2 is to the right although for square number 5 it is to the left even though these two squares are perceived identically.

(a) (b)
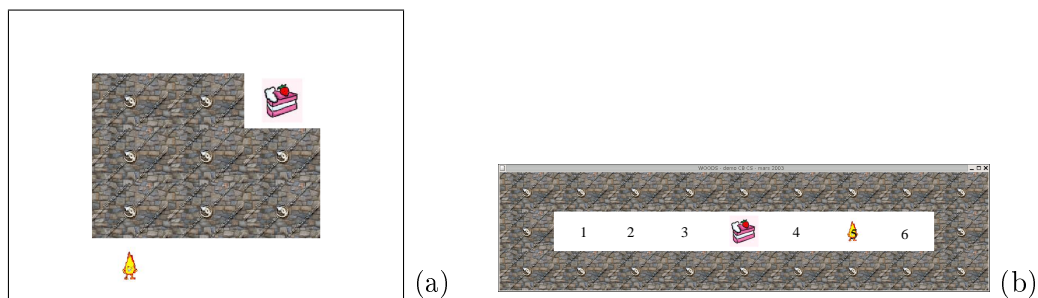
FIGURE 5.5 – Markovian Woods1 (a) and Non-Markovian Woods100 (b) Environments

The system learns by alternating between an iteration in exploration mode (selecting the action using the 'wheel of fortune' mechanism) and an iteration in exploitation mode (choosing the best action). The curves only take into account the results in exploitation mode, dealing with the average of the last ten iterations.

GEMEAU can deal with this two classical examples, it converges for the multiplexer (figure 5.6a) and for *Woods1* (figure 5.6b). For the multiplexer, we achieve a 100% success rate. For *Woods1*, we achieve solutions similar to the minimum number of movements. The results are conclusive: in both cases we reported performances similar to those described in the results of [Wilson, 1994].

Furthemore GEMEAU allows the rapid evaluation of derived classifier systems. We compared the ZCS and ZCSM results in the *Woods100* non-Markovian environment (figure 5.6). Our results redisover the ZCS' difficulties obtaining optimal rules in non-Markovian environments. They also confirm that our architecture can be used to extend the capacities of ZCS to non-Markovian environments.

---

[4] There are no perceptions values corresponding to different states of the agent
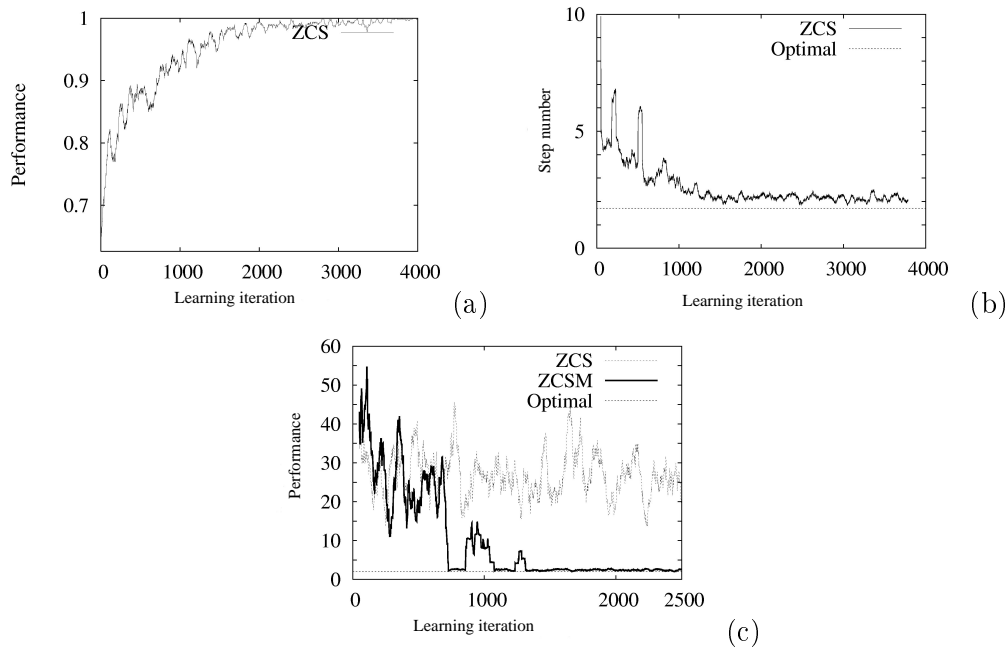
FIGURE 5.6 – ZCS multiplexer learning (a) and in Woods1 (b). As of a sufficient number of iterations, our system conducts the multiplexing function and obtains the minimum number of movements in the case of Woods1. ZCS and ZCSM learning in Woods100 (c).

5.5 An application example: stressed agents

Stressed agents are problem-solving distributed algorithms based on a behavioral metaphor of groups of humans. The algorithm, described in [De Loor and Chevaillier, 2003], was tested on the problem of resolving timetables. This is a distributed resolution technique making up a research heuristic similar to eco-resolution. A community of agents attempts to find a solution to the problem by sending requests, propositions, refusals or cancellations. The emission of these messages depends on a variable referred to as *stress*, whose evolution is itself defined by the perception of the messages and by two other variables: personal sensitivity and collective sensitivity. When stress exceeds critical thresholds (known as crisis thresholds and query thresholds), different decision-making mechanisms are triggered in order to partially modify the solution being calculated in order to direct the search. The crisis and query thresholds, and the personal and collective sensitivity of each agent, are empirically defined. Figure 5.7 shows an example of their influence on the algorithm's performance. The optimal values hinge on this issue and experimental study of resolution scenarios shows that their modification, during this period of resolution, could improve the algorithm's performance. We wanted to test this hypothesis using classifiers systems that should learn to modify these parameters, in order to adapt to different problems at different stages of the problem-solving process.
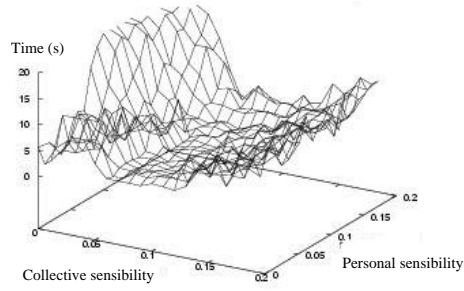
FIGURE 5.7 – The influence of the personal and collective sensitivity parameters on average resolution speed

5.5.1 Learning

In this example, the environment is most certainly non-Markovian and is highly dynamic. Indeed, an agent's stress synthesizes the entire history of its actions into one value. A given stress value very likely corresponds to a number of different results and just as many possible timetables. Dynamism, however, is rendered implicit by agentifying the problem. Within the context of this study, the most interesting aspect is the definition of the input interface and the size of the classifiers. Using the generic tool, we were able to test various criteria: *hypotheses concerning the input interface, different kinds of reinforcement* and *different kinds of classifiers systems.*

5.5.1 - A Hypotheses concerning the input interface

The problem was to find an appropriate perception that might facilitate learning. The following entries were used successively:

1. agents' stress,

2. the by-product of agents' stress,

3. the number of *crises* (stress exceeds crisis threshold) recently experienced by the agent.

Taking advantage of the possibilities presented by our architecture, a number of hypotheses were tested by *CS_II* inheritance:

▷ Different sizes for encoding the inputs (information from the environment is transferred into a symbolic binary representation by the input interface).

▷ Different combinations, in order to account for all or part of the perceived information.

The most advantageous solution was to take the three inputs into account, each of them being coded on 3 bits (8 thresholds).

5.5.1 - B   Different types of reinforcement

Two kinds of reinforcement were identified, either:

1. reinforcing during problem-solving, by giving a reward depending on the proximity of a solution, or

2. reinforcing once the problem is resolved, giving a reward depending on the problem-solving time.

It is this latter kind of reinforcement that we chose to use. Reinforcing during problem-solving posed a number of problems: firstly in terms of choosing the moment at which the reward would be given, and secondly, the reinforcement of activities leading to partial solutions do not necessarily reinforce the path to a complete solution.

5.5.1 - C   Different types of classifiers systems

We applied the following systems:

▷ Simple XCS, which have difficulty becoming stable. As the environment is non-Markovian, this provokes the system's instability.

▷ XCS with intermediary memories [Lanzi and Wilson, 1999], designed for non-Markovian environments, proved disappointing. We identified the interference present in the input interface, consecutive to the asynchronous and non-determinist aspect of the interactions between agents, as a possible emergent factor of these results.

▷ XCS designed for noisy environments [Lanzi and Colombetti, 1999]. This solution gave us more satisfactory results in terms of learning speed. However, the quanity of parameters required to characterize an XCS appears to be a real problem and undoubtedly an inconvenience in terms of their use.

▷ ZCS with or without memory which, in the end, gave the most encouraging results. It should be noted that these are also simpler, and that this simplicity does not inhibit performance.

5.5.1 - D   Results

The graphs in figure 5.8 (a) illustrate a ZCS's learning (using the configurations from [Wilson, 1994]), and figure 5.8 (b) shows the relative learning speeds of a ZCS and an XCS for this problem. We were thus able to test the flexibility of our tool in terms of adaptability and stability in terms of use at the heart of an open complex system.
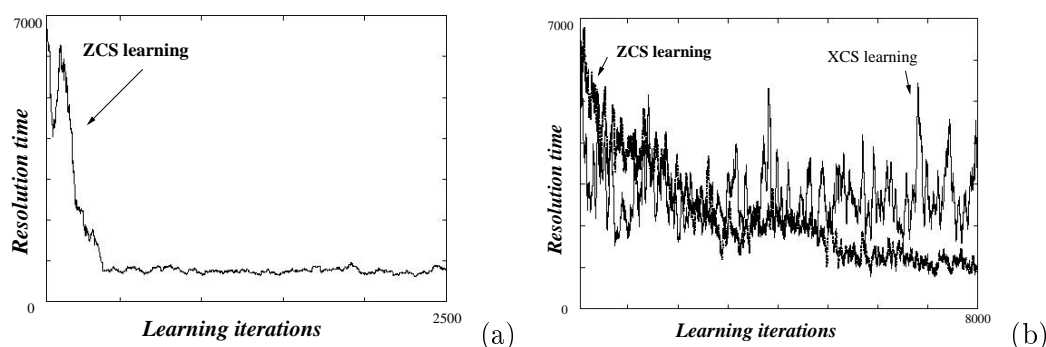
FIGURE 5.8 – Resolving the timetable problem using stressed agents with ZCS learning in one of the agents (a), and comparison between learning with XCS and with ZCS for a more complex problem (b)

5.6   Conclusion

Having described existing classifiers systems, we illustrated a more general classifiers system which groups together the traditional systems. We put forward one application enabling traditional systems and their variants to be both modeled and extended. This implementation is flexible enough to be used for a variety of problems as it proposes an interface between the environment and the classifier system (input/output/reinforcement). It has been used for different problems and to test many types of classifiers systems and different conceptual hypotheses quickly, as well as to obtain significant comparative results. Among other things, these tests showed us the interest of being able to access a library of classifiers systems with which we should be able to define a methodology for choosing learning algorithms based on certain stages of the tests.

We are currently studying the use of classifiers systems at the heart of a training platform which focuses on immersing the learner in the learning situation using virtual reality (project PEGASE see next chapter). A key element lies in the use of *pedagogic agents*, which are virtual entities that assist learning through particular educational roles (companions, troublemakers, advisors, etc.) in interaction with the learner. The main idea behind this research is to provide human instructors with a tool that enables them to specify the behavior of the pedagogic agents so that they might adapt their interventions according to the context of the simulation and the learners' errors and personal characteristics (abilities, curriculum, etc.). The instructor, who will be an expert in a domain but not a specialist in computing, will react intuitively, in a manner which is difficult to standardize. We suggest that, using classifiers systems, s/he should "teach" the pedagogic agents the relevant actions and reactions which would improve the learner's performance. In this context, the classifier system's input is the learner's actions, characteristics and errors, and the output corresponds to the educational strategies and reinforcements imposed by the instructor.

Contents of Chapter 6

Chapter 6

PEGASE — a pedagogical system for virtual reality environments

The context of this chapter is the creation of human learning environments using virtual reality. We propose the integration of a generic and adaptable intelligent tutoring system (PEGASE) into a virtual environment. The aim of this environment is to instruct the learner, and to assist the instructor. The proposed system is created using a multi-agent system. This system emits a set of knowledge (actions carried out by the learner, knowledge about the field, etc.) which PEGASE uses to make informed decisions Our study focuses on the representation of knowledge about the environment, and on the adaptable pedagogical agent providing instructive assistance.

6.1   Introduction

MANY fields of learning, like driving or professional training for firefighters, require learners to experience the setting in which they will work or operate. The learners must therefore acquire not only knowledge, but real hands-on skills. Virtual environments (VE) immerse learners in such situations. Figure 6.1 gives three examples of a road safety application (ARéViRoad) [Herviou and Maisel, 2006], a SEVESO plant application [Edward et al., 2008] and Gaspar for logistics on aircraft carriers [Marion et al., 2007].

This work is designed to teach decision-making in VE. Tutoring systems to instruct learners and assist instructors already exist [Lourdeaux et al., 2002, Rickel and Johnson, 1999c], but are dedicated to a specific VE. In this chapter, we propose an independent VE tutoring system called PEGASE, in the field of procedural and collaborative work.

Figure 6.1 – From left to right: screenshots from the ARéViRoad, Virthualis and Gaspar applications

6.2 Context: acquisition of human skills using virtual environments

Traditionally, most training programs aim to transmit *knowledge*. However, to facilitate the acquisition of knowledge, we must build on our prior knowlegde and skills. In this context, we propose the use of Intelligent Tutoring Systems (ITS) in which this knowledge is used in conjunction with the training setting. In this case, knowledge can be manipulated, e.g. to automatically question the learner. Being competent does not only mean having acquire knowledge, but also *being able to use that knowledge*. In order to facilitate the acquisition of knowledge, we must provide the learner with the right setting. To this end, we suggest using interactive systems by which the learners can be immersed in VEs in which they can make trial attempts, take initiatives, make mistakes, and try again in a similar situation (which may not be possible in reality). The simulation therefore provides an environment common to the learner, the instructor, and to the skill to be acquired. It mediates the learning relationship (learner-skill) as well as the instructive relationship (instructor-learner). Thus, computer-generated simulations, combined with an ITS, create an opportunity to improve learners' skills by associating knowledge with the possibility of putting their skills into practice.

ITS have already been used without being associated with virtual reality. As [Wenger, 1987] has shown, they usually conform to one of four models. The first, known as the domain model, contains a representation of the knowledge linked to the skill to be acquired. ITS also use a learner model which defines the learner's personal characteristics and ascertains the condition of the knowledge at a given moment. Using the domain and learner models, an ITS can evaluate the knowledge acquired by learners by comparing their activity with information about the field. However, the main objective of the ITS is to provide appropriate assistance to the learner or the instructor, depending on the setting (following activities or offering assistance). In this context, the pedagogical model can be used to make choices with regard to the training objective, with the aim of facilitating learning. Finally, an interface model is used to exchange information between the system and the user. Until now, this model has not been reified[1] in existing VEs designed for learning.

Within the context of our VE, we consider an ITS as a system which is part of the human

---

[1] Reification is a process through which concepts are explicitly represented by semantic representation (classes) to conceptual manipulation.

Virtual Learning Environment (VLE). We propose to evaluate the extent to which ITS are integrated within existing VLE. We have grouped VLE into three categories:

1. **VLE as conventional simulators**
   This first category includes those applications which include none of the four models, such as an application designed to assist in both maintenance and control of mobile cranes [Levesque, 2003]. In this kind of VE, the system provides no explanations about the task to be performed, which would require a domain model. The environment is therefore unable to adapt to the learner, as this would require a learner model. Finally, the teaching method is the instructor's responsibility. This sort of system is not able to make decisions regarding instructive interventions, however, it can help learners to improve or modify pre-existing skills.

2. **VLE with domain and/or learner models**
   This second category of VE is made up of applications which include a domain model and/or a learner model [Hubal, 2008]. The most well-known example of this type of VE is STEVE, a virtual character who assists in both teaching and learning procedural tasks [Rickel and Johnson, 1999c]. Using the domain model, STEVE can demonstrate and explain the procedure and above all, verify the learner's actions. However, STEVE intervenes on demand. He is incapable of knowing when, how and why to intervene, which would require a pedagogical model. In a system such as this it is possible to acquire skills, but the participation of the instructor is still required for all pedagogical interventions.

3. **VLE with domain, learner, and pedagogical models**
   This final category groups together the VEs presenting not only domain and learner models, but also a pedagogical model [Amokrane et al., 2008]. Let us examine the example of the educational agent, HAL, from the FIACRE system [Lourdeaux et al., 2002]. The application is designed to instruct individuals in learning to drive TGV high speed trains, using virtual reality (intervention on railways). As well as having all of STEVE's abilities, HAL assists the instructors in structuring the pedagogical discourse. In concrete terms, each anticipated behavior corresponds to a different instructive assistance (additional information, explanation of an object, etc.). The instructor must therefore list the possible errors for each piece of knowledge to be acquired. Furthermore, for each of these errors, the instructor must specify the way in which these pedagogical strategies should be conducted through instructive assistance, and furthermore must do so for each exercise. The main advantage of this kind of VLE lies in the assistance to the instructor in terms of the educational relationship linked to the learner, and in the didactic relationship linked to the skill to be learnt. However, the instructor must specify all of the knowledge to be acquired for each exercise.

Thus, most VLEs only include representation of the knowledge about one specific domain. Systems proposing a diagnostic component only rarely provide a mechanism for instructive assistance. HAL seems to us to be the most successful of these systems. However, the instructor must still make a list of the possible errors and specify the educational strategies for each exercise. Furthermore, the impact of the instructive assistance on the learner is not taken into consideration. In concrete terms, any proposed assistance which does not help the learner to make progress will be updated each time that specific situation occurs.

In order to resolve these shortcomings, we propose the integration of an intelligent tutoring system within a VE. This system must propose a flexible pedagogical model, i.e. a model in which instructive concepts can be easily added, modified or deleted. Furthermore, a model such as this must be *generic*, in so far as the pedagogical model must be exploitable independently of the task to be performed. Finally, the knowledge of the pedagogical model, along with its past experience, could be used to automatically suggest the appropriate interventions by taking into account both the learner and the context of the simulation: the system therefore becomes *adaptive*. Our model is called PEGASE (PEdagogical Generic and Adaptive SystEm).

In the next section, we will describe the global architecture of PEGASE. We will then go on to present our domain model (see section 6.3.1) and a description of our pedagogical model (see section 6.3.2), followed by a discussion of the advantages of our proposed models (see section 6.4). It must be noted that the proposal described here is applicable within the context of *the learning of procedural and collaborative tasks* and cannot be used in general learning situations.

6.3 PEGASE

Our proposal consists of reifying the four classic ITS models (domain, learner, pedagogical, interface), within a VE. We believe that errors can provide crucial information and thus decided to introduce a model called "error model". It is through the use of this new model that we will be able to generalize (something HAL could not do). Furthermore, we have added an "instructor model", in which the instructor specifies the knowledge about the exercise to be performed. The instructor defines the guidelines which describe the procedure(s) to be carried out, and the role(s) played by the learner (and consequently those which must also be activated automatically).

These models must provide solutions to counter the shortcomings of the existing systems described above and must therefore display two important characteristics: genericity and adaptability. We thus suggest that, it is possible to incorporate a generic and adaptive ITS from a VE by reifying the 6 ITS models.

So that each model share its information and conduct its analyzes autonomously (independently of both the situation and of other models), an autonomous entity (known as an agent) is associated with each model.

The agents interact by exchanging messages containing data (see Figure 6.2). This data can be extracted from the situation or inferred from the agent's internal reasoning using its knowledge (the model to which it is linked).

Step 1. *Observation*:

Using the interface model, the system analyzes the learner's activity. The elements that are important for learning are supplied to the learner model. This information concerns the learner's actions, those elements which the learner can observe, and the learner's movements.
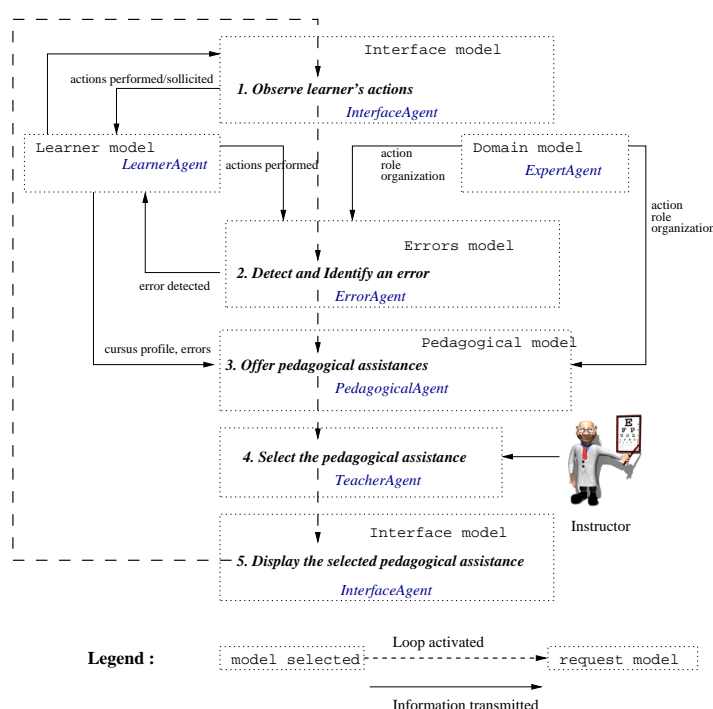
FIGURE 6.2 – The instructive process of our five-stage system.

Step 2. *Detecting and identifying an error*:
The system analyzes the learner's actions (learner model) and compares them with the actions to be performed (domain model). This comparaison is used in order to detect errors. If one is detected, an error identification mechanism is set up (using the error model).

Step 3. *Proposing instructive assistance*:
Using the learner model (characteristics, activities, errors, etc.), and the domain model (knowledge of the organizational structures), a mechanism stimulating instructive reasoning recommends the instructive assistance for the given situation. It must be noted that this step is not optional; it occurs even if no error is detected.

Step 4. *Choosing instructive assistance*:
The instructor can choose one specific instructive assistance amongst those proposed.

Step 5. *Representing instructive assistance*:
The instructive assistance selected is presented in the VE.

To use the information from the VE, we must inform the environment in order to obtain controllable knowledge. This creates an informed VE (see section 6.3.1). The environment will then be *reified*. This knowledge is complemented by additional information contained within the 6 ITS models. This data makes up a knowledge base for the pedagogical model which we call the *pedagocical situation*. This knowledge fuels the ITS's motor for making *instructive decisions* (see section 6.3.2). An example of the way in which the rules governing this motor are specified is presented in section 6.3.2 - C.

6.3.1 Domain model

To reify the concepts of the domain, we define the Veha (VE for Human Activity) meta-model. It describes the VE, not only in terms of geometric space, but by providing the semantics required for the artificial agents (ITS, autonomous characters) or humans (learners or instructors) to be able to construct for themselves a representation of the environment and act together to reach their goals. The `Veha` metamodel (M3) enables the construction of VEs models (M2) and the corresponding *concrete* VEs (M1) (see table 6.1). `Veha` is based on `Uml`[2]. It extends `Uml` because `Uml` does not define the specific concepts of virtual reality.

M4	`Mof`[3] (`Uml` limitation)			
M3	`Uml` metamodel	`Veha` metamodel		
M2	`Uml` *user model*	VE_1 model	...	
M1	*user object*	VE_{1a}	VE_{1b}	...	...

TABLE 6.1 – Layers of modeling (M*i*) : the position of `Veha` within the `Mof` framework, in parallel with `Uml`.

6.3.1 - A   The `Veha` metamodel

The ITS needs to know which objects make up the VE, how to access it, its properties, its behavior and how to interact with it. Three kinds of knowledge can be expressed using `Veha`:

1. Domain concepts: This entails the semantic description of the concepts relating to the field of activity concerned. It represents some of the knowledge that the learner must acquire (section 6.3.1 - A).

2. The possibility of structuring and interacting with the environment: These concepts resemble those suggested in *smart objects* [Kallmann and Thalmann, 1998] which reify those properties required for interactions. The means available to the learner or to the ITS must be specified in order to modify the environment (section 6.3.1 - A).

3. Entities' behavior: Within the framework of a VLE the environment's reactions to the learner's actions must be simulated. Entities' behavior also represents one of the elements of the knowledge to be transmitted and must be enforceable (section 6.3.1 - A).

In the following part of this section, we explain how `Veha` can be used to express these three kinds of knowledge.

---

[2] `Uml` (Unified Modeling Language) an object modeling and specification language (http://www.omg.org/spec/UML/).

[3] `Mof` (Meta-Object Framework) a meta-model used to formally define `Uml`.

Domain concepts

Knowledge of the domain is expressed both at the model (concept) level, and at the level of the occurrences of these concepts (tangible objects populating the environment). In Veha as in Uml, this knowledge is represented by classes (*Class*) and instances (*InstanceSpecification*).

In Veha, the notion of class is used to define a type of object (Figure 6.3) from domain-specific ontology. The aim is to be able to apply semantics to each of the business concepts, whether or not they are tangibly represented in the VE (concepts *vs* concrete objects). All classes stem from the *Element* class. This class enables the identification of each of the elements of a business model from its name and the addition of a textual comment. This can be useful when providing the user with explanations regarding the significance of an object.
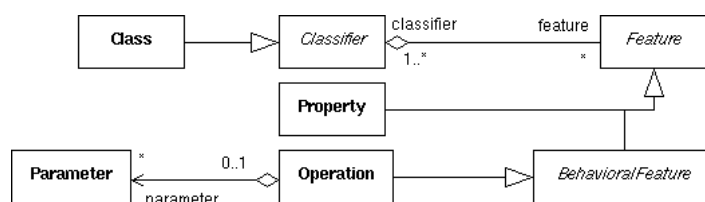
FIGURE 6.3 – Class diagram from the Veha metamodel: *Features* of a *Classifier*

The structural properties (Property) and behavioral features (BehavioralFeature) of the classes are assigned to the Classifier[4] *via* the Feature class. The Property class represents the structural component of the *Classifier* (as much the attributes as the relationships with other business concepts). As in Uml, the *Operation* class is the only tangible sub-class of BehavioralFeature. It is used to express the effect that an object or a user can have on another object. It does this by defining the object's actual behavior rather than the method used to achieve that behavior. The way in which the behaviors associated with Operation are modeled is described using behavioral models (see section 6.3.1 - A).

The Veha's second key concept is the notion of Class and Instance, synonymous for the object. The *InstanceSpecification*, Slot and *AssociationInstance* classes represent the instantiation of Class, Property and Association respectively. The term *InstanceSpecification* indicates that here, we represent an M1 level entity (see table 6.1), independently of the circumstances under which it is implemented.

The set of knowledge about the environment as specified in Veha can be accessed by the ITS and by the users (learners or instructors). The ITS can, for example, suggest to the learner a list of operations to be performed on one specific kind of object. Likewise, the instructor can modify the environment during the simulation by changing the attribute values of a tangible object.

---

[4] UML metamodel class which generalizes the concept of class.

The possibility of structuring and interacting with the environment

Most of the tangible objects within VEs are represented geometrically, and are situated within the environment. The learner must be able to observe, recognize and manipulate these objects. The ITS also needs to be able to manipulate them within the context of the instructive assistance that it will implement (transparency, refocusing from the learner's point of view, etc.). Knowledge about the geometry of these objects must also be specified so that the ITS will be able to recontextualize its suggestions within the VE. These objects are entities and all have the properties of the instances, i.e. `veha::Class` as well as geometric and topological properties (see Figure 6.4).
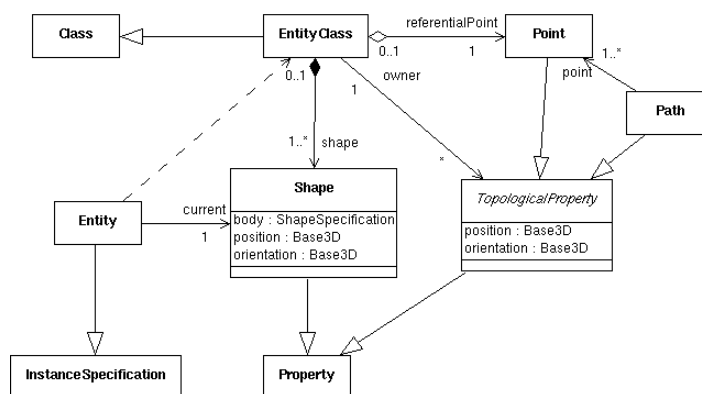
FIGURE 6.4 – Class diagram from the `Veha` metamodel: *EntityClass*.

Each entity is located at a global reference point. The *Shape* class is used to assign an instance of `EntityClass` to a graphical representation in the VE. It is possible to assign many forms to one class of entity. The ITS can use this knowledge to highlight an object or, on the contrary, to hide it. The *TopologicalProperty* class supports the notion of location (position and orientation) and is used to describe the topological properties of the elements within the VE. It is possible to assign informed points to an entity (`Point`) which can be used to create an interaction. This information is used by the ITS to turn the learner's attention to a specific object, for example.

Any entity within the VE is an instance of the `Entity` class, which derives from *Kernel::InstanceSpecification*. The values of an entity's properties are defined by its `slots`. So these depend on the semantic, morphological, geometric and topological properties of the objects within the VE (supplied by `InstanceSpecification`).

Entities' behaviors

When the learner carries out an action in the environment, that environment must react in a realistic way for the learner to be able to understand the consequences of his actions. The learner therefore constructs a representation of the entities' behavior. For the ITS to be able to regulate this representation, the knowledge of entities' behaviors must also be specified, as for the two previous kinds of knowledge, and it must also be enforceable.

The role of the `Behavior` package is to model the possible behaviors of the entities within the VE; the objective being for the model to be interpreted in real-time by a behavioral controller, and to be introspected online. As for the structural aspects, introspection relies both on the behavioral model (M2) and on its "instantiation", i.e. the way it is carried out (M1). The two classes which support these notions are `Behavior` and `BehaviorExecution` (see Figure 6.5). The `Veha` entities have reactive behaviors which are triggered by events that can be caused either by the learner or by another of the VE's entities.

Traditionally, behaviors are assigned pre-conditions and post-conditions concerning the entities and the environment. Behavioral modeling relies on state machines and the `Uml` activity model. Finally, it can also be based on functions written in programming language that can be consulted online (`OpaqueBehavior`). The first two methods are introspectable; the ITS can therefore describe or check the way the behavior is carried out.

The tutor can thus analyze, explain or check the context in which an entity's behavior is carried out by the learner. Better still, if a particular behavior has been specifically described (state machine or activity) it can also explain the way it will be carried out.
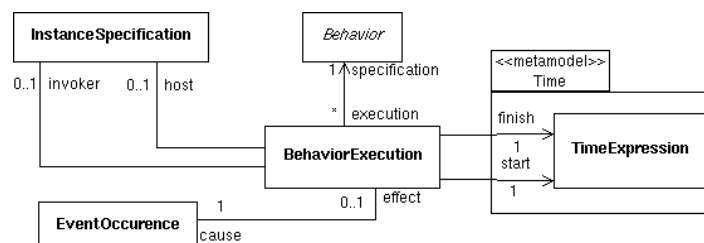
FIGURE 6.5 – Class diagram from the `Veha` metamodel: *Behavior::Common* package, the *BehaviorExecution* class

6.3.1 - B Example of an environment in `Veha`

The `Veha` metamodel can automatically interpret a model described in `Uml`. Figure 6.6 shows the class diagram for an example of a VE in `Veha`. This example comes from an application created in `Veha`, but which has been greatly simplified for demonstration purposes. The application (GASPAR, [Marion et al., 2007]) is made up of around fifty classes and more than one thousand entities. This model shows the classes `Deflector` and `CatapultCabine` (left window). The catapult cabine shields the operators working on the catapult deck of an aircraft carrier. A pod can open (raise above the deck) or close (drop back down into the deck). The business model specifies all of the pod's properties (`height`, `speed`, etc.). The reactive behavior of a pod is specified by a state machine (top right-hand window). This state machine is sensitive to the signals `Open` and `Close`. Therefore, when the pod is `Closed`, if it receives the signal to `Open`, it changes to the `Open` state and performs the operation `Open()`. Within the context of this application, this operation is described in detail by an `OpaqueBehavior`, a C++ code which carries out the visual displacement of the pod depending on the speed attribute, and updates the height attribute.

In much the same way, deflectors also react sensitively. Due to the additional needs of
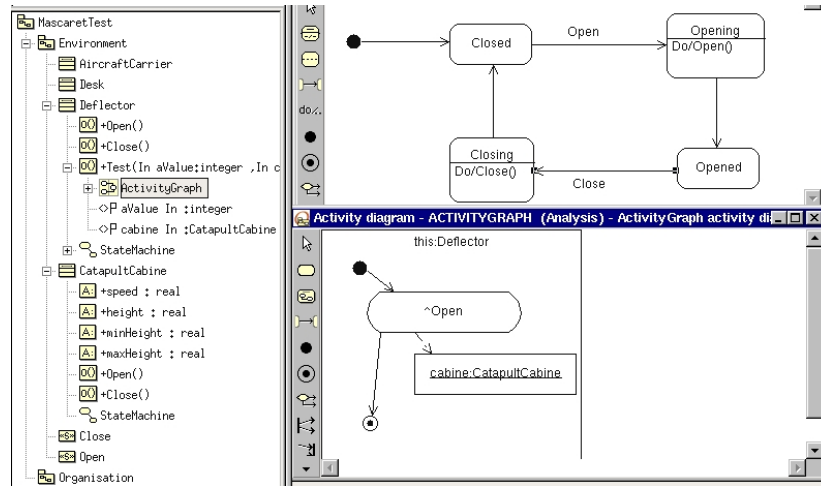
Figure 6.6 – Class diagram for a deflector and a catapult control pod.

this demonstration, we added a testing operation (`Test`). This operation takes its settings from a catapult control pod. The behavior of this operation is specified by an activity diagram (bottom right-hand window). Therefore, when a `Test` operation is evoked in an instance of the `Deflector` class, the operation sends the signal `Open` to the predefined pod.

This model is defined using *Objecteering* modeling software. It is then exported in an XMI file. The first proposal is to add an interpreter to the `Veha` metamodel within the *AReVi* virtual reality platform. The interpreter reads the XMI file and, for each class of `Uml` metamodel, creates an instance of the corresponding class in the `Veha` metamodel. Thus, for each business class defined in the XMI file, the interpreter creates a new instance of the `class` class from the `Veha` metamodel. In the context of our example, an instance of the `Class` class is created for the `Deflector` class, and another created for the `CatapultCabine` class. The interpreter enables the reification of the business model and provides an set of methods facilitating the introspection of this model. It is therefore possible to ask the interpreter for the set of a class's properties, the signal which enables the passing from one state to another, and the operation which will then be conducted, all independently of any tangible object.

The VE is populated with entities, the instances of the `Entity` class of the `Veha` metamodel. From a technical standpoint, these instances are defined in an XML file. Using `Uml`, class instances can also be described and exported in the XMI file. However, no `Uml` modeler can make it simple to attribute a shape and a position to these instances. The geometric design of the VE is, in general, the result ouput by specialist modelers such as *3DS MAX* or *Blender*. We therefore suggest using an export `plugin` for *3DS Max* which would generate the instance file read by the interpreter. Figure 6.7 shows the visual result of the file defining the model (XMI) and the instance file (XML) in an application implemented using ARéVi. The interpreter also provides the methods for interrogating and manipulating the entities. It is therefore possible to ask an entity for its property values, to carry out an operation, or to send it a signal in order to change its state.
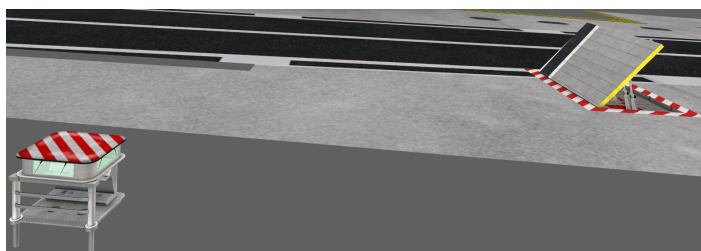
FIGURE 6.7 – Visualisation of the instances of `CatapultCabine` and `Deflector`.

6.3.1 - C   Procedure and collaboration

Here we examine the acquisition of skills. The domain model not only contains knowledge about the environment in use, but also knowledge about the task which must be performed within that environment by the learners. Within the context of this research and the examples given in the introduction, activities are defined by the procedures describing the `Action`s to be performed by a number of entities, each with specifically defined roles. We use the same assumption as for the environment and propose the use of a metamodel based on `Uml` in order to define these activities. The procedures are therefore defined by activity diagrams. This kind of diagram uses the traditional possibilities for organizing its `Action`s (parallelism, sequence, junction, condition, etc.) As we are dealing with representing human activity, we consider that the sequence of activities takes place in an asynchronous manner.
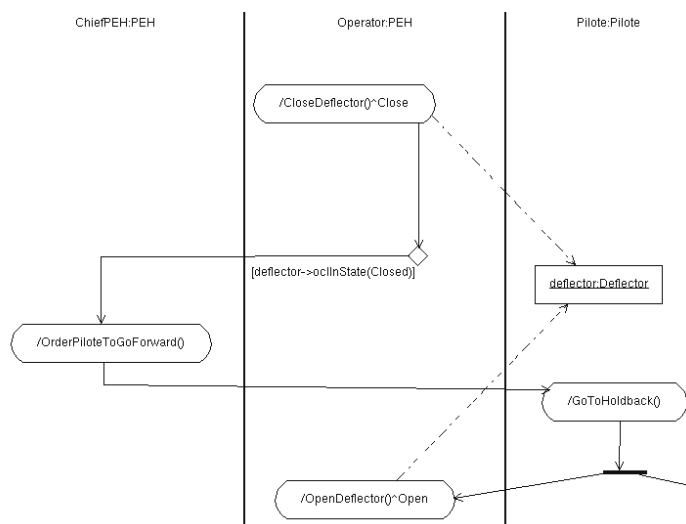
FIGURE 6.8 – Example of a procedure written using an activity diagram.

The organization roles are represented by activity corridors. The name of the corridor defines its role and its type, as well as the type of agent that is authorized to take this role. As in `Uml` 2.1, there are many different types of activity. This could be the execution of an agent's operations, a basic virtual action (playing an animation, reaching a given position, etc.) or sending a signal to a specific resource. The resources are drawn on by the environment's

entities and represented by objects in `Uml`. The conditions are expressed in `Ocl` and stem from the roles and resources participating in the procedure. Figure 6.8 illustrates the example of a procedure expressed using an activity diagram. This procedure solicits the intervention of three roles (such as `Operator`) which must be played by characters of a pre-defined type (`PEH` for example). The characters which play these roles are those which are effectively instantiated in the environment. This procedure aims to make the airplane which is to be catapulted advance towards a given point by manipulating the deflector (a protective plate). The example of the procedure in Figure 6.8 illustrates the complementary nature of the state machines used to define the reactive behavior of the objects in the environment and the activity diagrams defining a procedure. A procedure's action can be represented by sending an event to a given object to be manipulated, and the conditions of moving onto the following action can depend on the current state of the object.

We implemented agents' behaviors using knowledge about the procedures to select their actions. The learner plays one or more roles in the context of these procedures. The ITS also draws on this knowledge in order to choose which assistance to suggest. As for the environment, there are two levels of modeling available to the agents (including the ITS) and the users (instructors): the organizational structure and the organizational instances. The intelligent tutor is therefore able to recognize the sequence of actions independently of all organization. It can also follow the precise progress of the procedure being carried out in the team in which the learner plays one or more roles. It is therefore able to detect the learner's errors with respect to the order of the actions to be completed and compliance with the conditions defined in the procedure [Thanh Hai et al., 2009].

6.3.2 Pedagogical model

Knowledge about the environment (the entities and about the task to be performed) are represented with the `Veha` model. Our ITS can thus manipulate them in order to construct its own knowledge, as shown in (section 6.3.2 - A), and can simulate pedagogical reasoning 6.3.2 - B). Finally, a tangible implementation of the ITS is proposed in section 6.3.2 - C (specification of the rules of simulated pedagogical reasoning).

6.3.2 - A Pedagogical situation

It must be emphasized here that our work is done in the context of *in situ* learning. Within this theoretical framework, the contextual elements are paramount in the ITS's decision-making [Turner, 1993, Pomerol and Brézillon, 2001]. In our case, we refer to context as the pedagogical situation which serves as a basis for decision-making. The aim is to define this sort of context from a "generic" standpoint, which would enable us to alter information without having to take into account the specific task being carried out. To do so, we must separate knowledge about the task to be performed (see section 6.3.2 - A) from knowledge about the learner (see section 6.3.2 - A).

Information concerning the task to be performed

We positioned our work in the context of training for procedural work. The aim of the ITS is to assist learners in their progression through the procedure. The skill to be acquired relates to the completion of the procedure in a dynamic environment.

First of all, we can consider the procedure as a sequence of actions defined by an expert. The elements to be considered are therefore subject to *sequencing* which cannot be questioned, and sometimes cannot be explained. Secondly, we think that memorization of the sequence of actions could be facilitated through understanding. In this context [Richard, 1990] suggests adding the notion of sub-objectives to the procedure. To meet this aim, i.e. the completion of the procedure, a set of causally linked sub-objectives must be conducted. The procedure must therefore be studied taking into account the distance to the procedure's goal from a *causal*, rather than a chronological standpoint.

The above analysis highlights two ways of dealing with procedural learning: the study of business sequencing links which are strongly linked to the roles in the procedure, and the study of causal links between sub-objectives:

1. *Sequencing links*
   Sequencing links conduct the relationships between the actions using the strict description of the procedure. They are the direct consequence of the sequencing of actions as defined by the expert. We are interested in the information linked to the actions closest to the action requested by the learner. More precisely:

   ▷ the last correct action completed before that which the learner has just solicited;

   ▷ the action which has just been solicited by the learner;

   ▷ the correct actions to be carried out, taking into account the role(s) to be played (which are potentially different from the solicited action);

   ▷ the correct actions to be performed, when considering that all roles are played by the learner; and

   ▷ those actions following all the correct actions.

   We chose the actions closest to that solicited in order to try to reduce the "distance" between the goal (the end of the procedure) and the learner's location in the procedure. Technically, this is done by carrying out plan recognition based on the `Veha` activity diagram shown in section 6.3.1 - C. The pedagogical situation thus retains the knowledge linked to the actions that are chronologically close to that which is requested.

2. *Causal links between sub-objectives*
   The procedure can be considered like a graph representing the sequence of causal sub-objectives. We therefore are looking at all of the actions linked to the one the learner is performing. In concrete terms, this means the actions requiring the effect of the correct desired action (usage conditions, state of a resource, etc.). A distinction must be made between these links, which correspond to individual logic, and sequencing, whose links correspond to the organization of a collective procedure. Technically, we are dealing with the links between post-conditions and pre-conditions mentioned in section 6.3.1 - A.

It must be stressed that our objective here is to extract knowledge relating to the work to be carried out in order to assist pedagogical decision-making. Within this context, we look at the knowledge described in table 6.2. All the actions which have been identified up to this point (sequential and causal links) make up the pedagogical situation. More specifically, we are interested in the information related to the selected actions. At this point, we must specify the knowledge relating to the concept of action. From this perspective, the "action context" is made up of knowledge that is directly linked to the `Action` (description, resources, etc.), knowledge relating to the `Operation`, which is the target of the `Action`, as well as knowledge relating to the agent that has requested the action, since that agent is the central character. We therefore use *action contexts* in order to represent the knowledge associated with particular actions (a sub-group of the environment made up of the entities and agents considered relevant in the context of the action).

Knowledge	Nature	Description
① Context of the previous action	Sequential	The last correct action to have been performed. This action serves as a point of reference from which one can position oneself in the procedure.
② Context of the requested action	Sequential	The requested action. This action could be correct or incorrect. The action has not necessarily been performed, in accordance with the pedagogical model.
③ Context of the correct action(s) without considering their roles	Sequential	In considering the last correct action, we can determine the actions to be performed within the context of the current procedure.
④ Context of the correct action(s)	Sequential	A sub-group of the previous item which does not take the roles played by the learner into account.
⑤ Context of the following action(s)	Sequential	For each correct action, we determine the actions which follow it according to the current procedure.
⑥ Context of related action(s)	Causal	In considering the actions to be performed following the last correct action, we retrieve the "causal" links between the actions independently of the procedure. We therefore obtain the actions which are related.

TABLE 6.2 – The pedagogical situation: knowledge about the task to be performed.

It is the responsibility of the pedagogical agent to construct this set of knowledge. The pedagogical agent retrieves or constructs the knowledge required about the task to be performed when it receives a message from the interface agent detailing an action which has been requested. This choice is debatable and indeed another possible solution is to update the knowledge when an error occurs. We chose to reconstruct the knowledge of the actions in order to retain the option to intervene, even if the learner's actions are correct. This means that we can provide pedagogical assistance in order to reassure the learner about the decisions that they have made, or conversely to imply doubt if it looks like they are about to make a mistake (e.g. confirming false rules which contradict the choices the learner has made).

Information concerning the learner

The information about the learner comes from a number of sources, but all of it is collected by the learner model. This information relates both to static data (such as age) and dynamic

data (such as elements of memory at a given time).

It should be noted that the learner's errors are recorded and are analyzed. Our error model is based on the Cognitive Reliability and Error Analysis Method (CREAM). This approach proposed a classification scheme which makes a distinction between observations of errors (phenotypes) and its causes (genotypes). The causal links between phenotype and genotype are represented using a number of consequent-antecedent links. Finally, the pattern could be associated with a method of retrospective analysis (the search for causes). The most probable cause-effect links is found using Dempster-Shafer's theory presented in [El-Kechaï and Després, 2006]. Similarly, the contexts relating to the actions are also recorded. This information allows us to see whether or not learner has already used a particular resource, for example. In concrete terms, we have just defined the input information and the relevant elements from which pedagogical decisions can be made.

6.3.2 - B   The pedagogical agent

The pedagogical situation (section 6.3.2 - A) gives us the option of triggering pedagogical assistance relating to the elements detailed within it. It thus provides the possible outcomes of the pedagogical decision-making process. We now go onto define a model to simulate the behavioral decision-making of the pedagogical agent providing instructive assistance, i.e. a model linking knowledge and the proposed assistance. It must be noted that we are working within the context of learning procedural and collaborative tasks. We must therefore consider:

▷ The atypical nature of the knowledge involved (knowledge stemming from basic pedagogical methods to virtual reality).

▷ Adaptability (the agent's reasoning processes must self-adapt in order to take past experience into account).

▷ This reasoning must be specified prior to the event (initial specifications can therefore be made by an instructor).

The criteria which arise from these considerations are as follows: expressiveness, hierarchy, modularity, reactivity and adaptability.

After examining the existing families of behavioral architecture (connectionist, automata-based, rule-based), we opted for the rule-based families which best respond to the criteria outlined above. More precisely, we chose classifier systems. This is a reactive and adaptive form of architecture, based on conditional rules.

We propose the use of a model based on a hierarchical classifier system, founded on the GEMEAU model described in the previous chapter. This system organizes knowledge while taking the abstraction of the data involved into account. It structures knowledge according to three levels, from rules based on abstract knowledge of educational methods (the pedagogical approach), to the rules based on concrete knowledge of virtual reality (pedagogical techniques), via an intermediary level (pedagogical attributes).

Each level of abstraction contains sets which group together a number of rules. One set represents a way of dealing with a particular approach, attitude or pedagogical technique. The rules are conditioned by the elements of the pedagogical situation, and favor the sets from the lower level. The system therefore uses a diffusion mechanism on all three levels which considers the rules matching the pedagogical situation. This gives rise to a list which then arranges the different suggestions for pedagogical assistance.
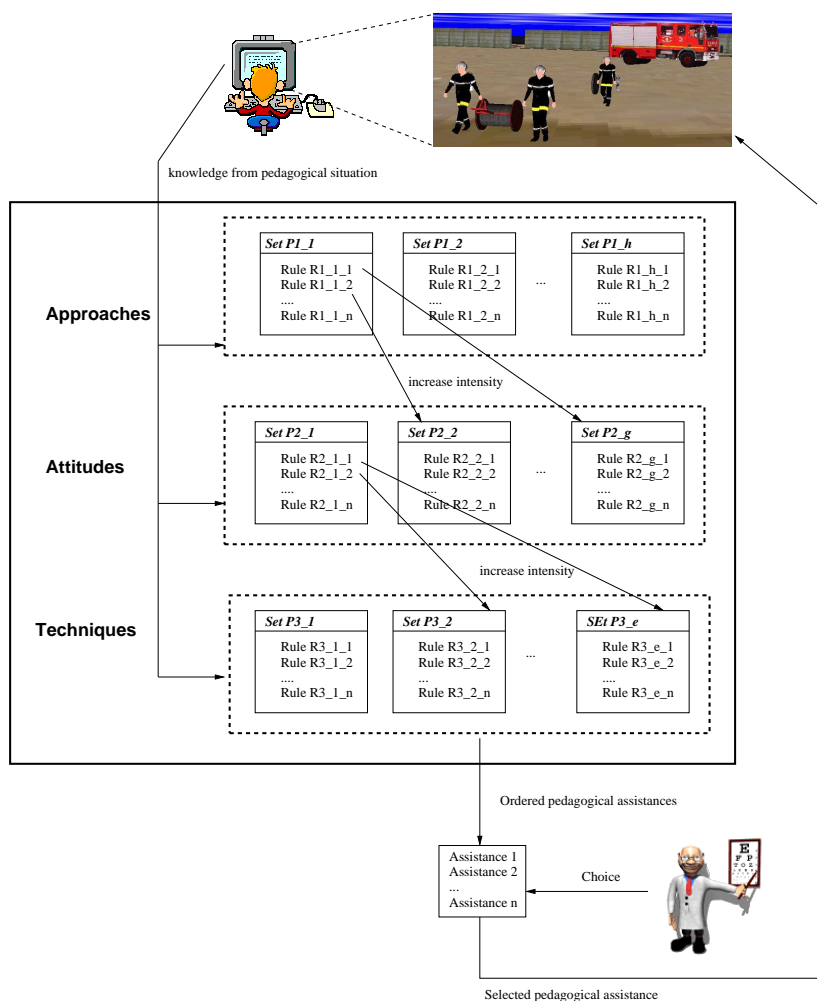
FIGURE 6.9 – Complete Representation of the Pedagogical Model.

Figure 6.9 illustrates the structure and the dynamics of the pedagogical model controlling the pedagogical agent's behavior. The information taken into account in the conditional part of the rules is retrieved by our ITS (pedagogical situation). These "inputs" are available at the three levels of data abstraction (approach, attitudes and pedagogical techniques). The rules whose conditional elements are satisfied in terms of input favor some of the sets of pedagogical rules from the lower level. The upper level (techniques), directly favors those pedagogical suggestions which can be applied within the environment. These suggestions are made to the instructor who chooses the one considered to be the most relevant.

Simulating pedagogical reasoning has two advantages:

1. As instructors are not always teachers, they too are being given pedagogical assistance.

2. The instructors are not simulation software experts, so the pedagogical agent will offer assistance to the learner who will have the opportunity to make the most of the VE.

6.3.2 - C Specifications of the pedagogical model

In order to implement the pedagogical model, the teacher must specify:

1. The sets of rules for the three levels of abstraction.

2. The pedagogical rules for each of the sets of rules.

Here, we will discuss information from the literature which can be used when specifying the pedagogical model.

Specifying the sets of pedagogical rules

We worked from the studies by [Lourdeaux et al., 2002] in order to define the sets of pedagogical rules. We obtained the following tables; 6.3, 6.4 and 6.5 corresponding to the three levels; approaches, attitudes and techniques, respectively. This information provides an opportunity to specify sets of rules at each of the three levels (see Figure 6.10).

Specifying the pedagogical rules

Once the sets of pedagogical rules are defined, the teacher must specify the associated rules.

A rule is represented by a sequence of characters. The effect and condition parts are based on the elements of the pedagogical situation.

In the following example, we position ourselves at the *Pedagogical Methods* abstraction level, with a set of rules called *Active*. The first rule for this set is fulfilled if the learner is a novice (`Learner.Level==novice`), if they have performed an organization error (`Learner.Error.type==procedural`) and if the action performed is different from the correct action (`!Task.RequestedAction in Task.CorrectActions`). In this case, the rule favors the `Explain` set from the following level.

```
if (Learner.Level == novice &&
    Learner.Error.type==procedural &&
    ! Task.RequestedAction in
      Task.CorrectActions)
then (Explain)
```
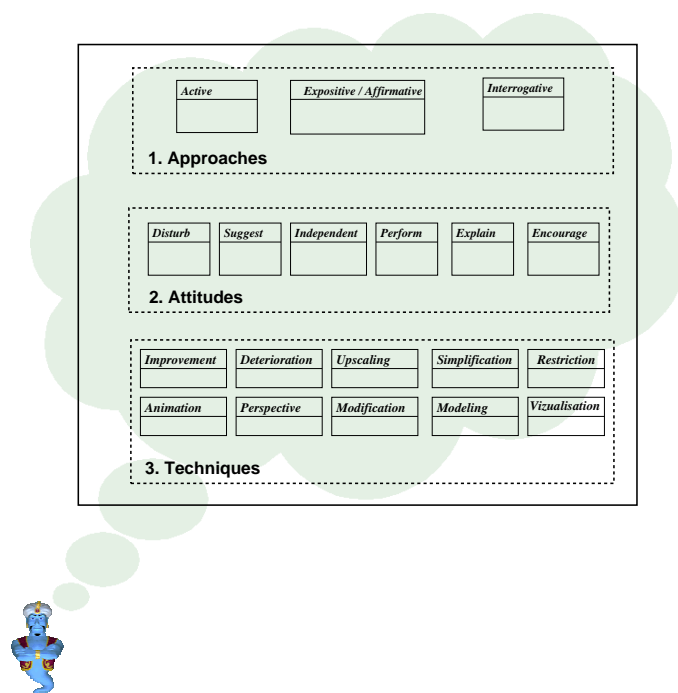
Figure 6.10 – Specifying the three levels of the pedagogical decision-making model.

Pedagogical Approach	Description
Active / Constructivist	An active approach is learner-centered, considering them to be the main actors in the learning process. This approach suggests techniques through which they can produce, create and search. The knowledge required can be found in the environment.
Expositive / Affirmative	This is the most traditional approach which uses the display technique. It is based on a content-transfer approach. Knowledge is external.
Interrogative	This approach makes recommendations to the learners, guiding them towards the desired outcome. Learners may have the impression that they have discovered something new, but it is the instructor who will have guided the thought process. Knowledge is internal.

Table 6.3 – Examples of set definitions for the "pedagogical approach" level of abstraction based on [Lourdeaux et al., 2002].

Pedagogical Attitudes	Description
Perform	Perform the task in the place of the learner. This strategy can be used by the instructor to show the learner the correct technique or move.
Disruption	Some instructors tease and disrupt the learners by giving them incorrect information or potentially incorrect solutions in order to test the learners' conviction of their ability to reason independently.
Suggest	Showing where the learners can find theoretical information or where to find information within the environment. These attitudes allow the instructor to show the learners that they can find the required information independently and therefore deal with the situation in a calm manner.
Independent learning	This attitude encourages the instructor to remain in the background as an observer rather than to intervene.
Explain	The explanations and information are also designed, quite simply, to explain the functioning of certain devices, rules of analysis, safety rules, etc.
Encourage	Encouraging the learners when they perform a task correctly.

TABLE 6.4 – Examples of set definitions for the "Pedagogical Attitudes" level of abstraction based on [Lourdeaux et al., 2002].

Pedagogical Techniques	Description
Improvement	Addition of visual and audio symbols or animated films.
Deterioration	Unrealistic images. (points of reference erased, feed-back, deteriorated proprioceptive elements, altered colours, blurred background/surround, reduction of objects, iconization, etc.).
Upscaling	Exaggeration of reality (representing objects on a larger scale, or that are surreal, brighter or shinier, etc.).
Simplification	Simplification of the virtual scene (a crowd can be represented by people with simplified movements, simplified objects, simplified kinetic systems, wireframe images, etc.), schematic representations of certain devices.
Restriction	Limitation of certain movements or actions (limiting the area within which the learner can move around, etc.)
Animation	Animated sequence (automatic positioning, keys which turn automatically once in place, etc.).
Perspective	Altering the learner's normal viewpoint (view from behind, above, etc.).
Modification	Changes in appearance and texture (colours, flickering objects, etc.).
Modeling	The representation of abstract concepts, of physical phenomena invisible to the naked eye, types of errors, etc.
Visualisation	Hidden mechanisms (the inside of a motor, gears, etc.).

TABLE 6.5 – Examples of set definitions for the "Pedagogical Techniques" level of abstraction based on [Lourdeaux et al., 2002].

Use

A specific pedagogical model was created from the structure described above, and from chapters by [Lourdeaux et al., 2002]. These sets are described in Figure 6.10, with each set containing an average of five rules. This pedagogical model was applied to two distinct VEs designed for learning collaborative procedures. No modification of the pedagogical model (sets and rules) is required for either of these applications, which although very different, are both based on the same kind of learning. However, we believe that these changes would only need to be made at the intermediate level. For other types of learning (for example for a scientific practic), these rules would probably need to be changed.

6.3.2 - D Artificial learning

Thanks to artificial learning, the weight of the rules for adapting to the instructor's preferences can be refined and their expertise imitated.

The learning algorithm is inspired by the *Bucket Brigade* [Holland, 1986, Wilson, 1987]. This system distributes remunerations to the rules which enabled them to be obtained. It is adapted to classifier systems with a list of rules which, when followed one after the other, lead to an action. In our case, this sequence of events corresponds to the passing from one level to another. Remuneration is reflected by the instructor's choice: the pedagogical technique which they choose defines the rules in the third level which will be compensated. By back-chaining, the rules in levels one and two are also compensated. The weights of the rules which match the *pedagogical situation*, but which participate in activating a technique other than that chosen by the instructor will decrease. The algorithm shares out the remuneration, including a tax which means that the rules which rarely match are not put at a disadvantage, and that the strong rules are penalized in order to retain the adaptive nature of the system.

Therefore, as the exercises progress, the pedagogical agent must make suggestions which correspond more and more closely to the instructor's decisions. The pedagogical agent could therefore temporarily take over and directly apply the assistance that it has chosen itself, should there be more than one learner at a time.

6.3.2 - E Use case : Gaspar

Gaspar is a virtual reality application developed to simulate human activities on an aircraft carrier. In Gaspar, a typical scene such as that shown in Figure 6.11 is made up of around 1000 entities, each with 3D representation (VRML), i.e. a total of 1 million facets. In this scene, there are around 50 agents, divided into 10 teams, each with an average of 5 roles. Each of these teams is responsible for an average of 5 procedures. The most complex procedure activates 9 roles and organizes 45 actions. In this scene, at each moment, around 50 behaviors are activated (both NPCs and entities). This sort of scene is implemented using *ARéVi* [5] and is simulated in real-time (around 40 frames per second) on a desktop computer with 2GB of

---

[5] http://sourceforge.net/projects/arevi/

FIGURE 6.11 – View of a scene on an aircraft carrier in GASPAR.

RAM, a 64 bit processor running at 1.3 GHz, and a GeForce card with 1GB of video memory.

The decisional behavior of the ITS relies on a classifier system in which each rule presents a set of conditions required to activate an educational *method*, *attitude* or *assistance*. The main advantage here is that the rules are formulated in a general way, at the M2 level, and deal with the data from the concrete environment (M1 level). The ITS knows how to evaluate rules such as: *"IF the entity is not in the state required to carry out the correct action and if the learner is novice THEN simplify the environment"*. Rules such as these can be expressed using the `Veha` metamodel, independently of the model of the virtual environment.

The veracity of these conditions is evaluated by the manipulation of the model, contextualized for specific environments using M1-level knowledge. For example in GASPAR, if the correct action is *tensioning the hook* in the context of the procedure *catapulting the Hawkeye aircraft*, the previous condition rule is automatically contextualized to *"IF the Hawkeye aircraft is not in the state launch bar down required to carry out the operation tensioning the hook"*.

The classifier system builds up a list of proposals for educational assistance made up of the action elements of activated rules. The assistances are evaluated by the manipulation of the model, contextualized for specific environments using M1-level knowledge. For example, the assistance: *"simplify the environment"* translates to a corresponding solution proposed to the instructor *"make transparent all entities except the Hawkeye aircraft"* (see Figure 6.12).

6.4 Conclusion

Before concluding, we would like to discuss the benefits of our proposal. The study described in this chapter began by examining previous studies in this field and analyzing the uses of pre-
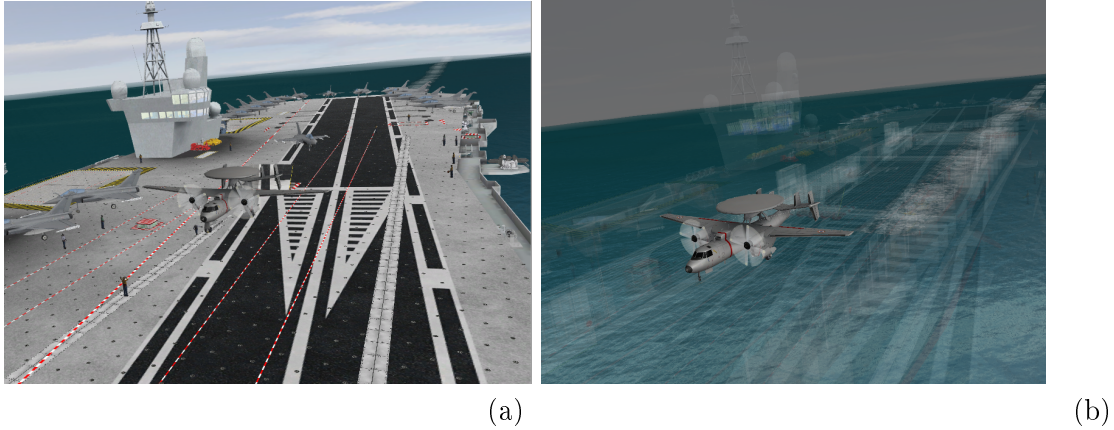
(a) (b)

Figure 6.12 – The effect of applying the pedagogical assistance "simplify the environment" in the Gaspar application. (a) before ; (b) after.

existing ITS within VLE. We then went onto show that the Hal system is the most successful, and highlighted the elements which could be improved. Indeed, in this system, the pedagogical model depends partly on the exercise, and the errors and pedagogical strategies must be defined. Furthermore, the instructor can only choose between two pedagogical methods (active or explanatory). We believe that it is possible to resolve the pedagogical model's problems of genericity and modularity.

Without re-examining every element of our work, we can show how our proposal could solve some of the difficulties of existing models. The knowledge used for pedagogical reasoning does not depend on the specifics of the task to be performed. Therefore pedagogical rules do not and indeed do not need to consider specific information, ("if the learner can see airplane 2 then..."), but will rather use general knowledge independently of the exercise ("if the resources of the correct actions are visible, then..."). In much the same way, although the pedagogical assistance proposes tangible solutions to the instructor ("make the fireman flicker"), generic knowledge is also manipulated independently of the exercise ("make the characters involved in the following actions flicker"). Thus, the genericity of our proposal is one of its strongest characteristics, as illustrated by the inclusion of our ITS at the core of numerous applications: learning of collaborative procedures on aircraft carriers (Gaspar) [Marion et al., 2007] and for firefighters intervening in Seveso high risk areas (SécuRéVi) [Querrec et al., 2004]. In addition, the pedagogical model of our ITS has strong modularity, as it offers the option of adding, deleting or modifying each of its components that participates in pedagogical decision making (rules or sets of rules). The pedagogical model uses the Gemeau model described in the previous chapter. Moreover, the artificial learning mechanism adapts the proposed pedagogical assistance to the learner-instructor pair. Therefore, our proposition provides solutions for the problems raised in the introduction. Finally, it must be emphasized that Pegase is directly based on the learner-instructor relationship.

However, we must not forget that there will undoubtedly be limitations linked to the use of our ITS in contexts of non-procedural learning. To be able to deal with this kind of training, we would have to rethink the elements which are so strongly linked to the notion of procedure, i.e. knowledge about the pedagogical situation.

Contents of Chapter 7

Chapter 7

Conclusion

"Do you believe the world to be inert, heavy, mechanical, substantial and dead? Do you believe the world to be real? We know that the bat, the whale and the human live in entirely different worlds. We know that this chair, this sheet of paper, this tree, this body, are also electromagnetic clouds vibrating in the emptiness with only the shape, colour and meaning that we attribute them. We know that the smallest of our thoughts influences our actions, that our actions influence our perceptions, that our perceptions influence our thoughts and that our fragile existence gives rise to this unstable whirlwind. For consciousness, feelings, desires, imagination, inferences, perpetual and multiple discourse, logic, delirium, quest for meaning and conscience, I tell you this: the world is virtual".

PIERRE LÉVY, LA MONTÉE VERS LA NOOSPHÈRE

IN this accreditation to direct research, our work on adaptive behaviors for entities in participatory virtual environments has be presented. This chapter is the opportunity to summarize our study (section 7.1), to discuss the provisions on our suggestions (section 7.2) and to envisage future research (section 7.3).

7.1   Summary

Chapter 2 presented the CHAMELEON project, a probabilistic behavior model and imitation learning algorithm for believable characters in video games. In some video games, humans and computer programs can play together each one controlling a virtual humanoid. These computer programs usually aim at replacing missing human players but can be spotted as being artificial by players. Our objective is to find a method to create programs which behavior cannot be told apart from players when being observed playing the game. We call this kind of behavior a believable behavior. To achieve this goal, we choose a model developed by Le Hy to generate the behaviours by imitation. The model uses probability distributions to find which decision to choose depending on the sensors. Then actions are chosen depending on the

sensors and the decision. The core idea of the model is promising but we propose to enhance the expressiveness of the model and the associated learning algorithm. We first revamp the organization of the sensors and motors by semantic refinement and add a focus mechanism in order to improve the believability. To achieve believability, we integrate an algorithm to learn the topology of the environment. Then, we revamp the learning algorithm to be able to learn much more parameters and with greater precision at the cost of its time of convergence.

Chapter 3 described the JABU project, a juggler with anticipatory behavior in a virtual universe. This chapter deals with the framework of believable behavior simulation. To be believable, virtual entities must be provided with the ability to ancitipate, so that they might predict the behavior of other entities and the consequences on the environment. In order to do so, we propose an original approach where the entities possess an autonomous world of simulation within the simulation, in which it can simulate both itself (with his own model of behavior) and the environment (with the representation of the other entities' behavior). This principle is illustrated by the development of an artificial juggler in 3D. For this application, the juggler predicts the behavior of the balls in the air and uses its predictions to coordinate its behavior in order to juggle. In addition, an interface allows the human user to launch additional balls that the juggler must intercept and add to the juggling.

Chapter 4 detailed the CAMUS project. It defined the use of fuzzy cognitive maps in simulating individual adaptive behaviors. This chapter focuses on the simulation of behavior for autonomous entities in virtual environments. The behavior of these entities must determine their responses not only to external stimuli, but also with regard to internal states. We propose to describe such behavior using fuzzy cognitive maps (FCM), whereby these internal states might be explicitly represented. This chapter presents the use of fuzzy cognitive maps as a tool for specifying and controling the behavior of individual agents. First, we describe how fuzzy cognitive maps can be used to model behavior. We then present a learning algorithm allowing the adaptation of FCMs through observation.

Chapter 5 described the GEMEAU project, a generic model for experimenting and using a family of classifiers systems. Classifiers systems are tools adapted to learn interactions between autonomous agents and their environments. However, there are many kinds of classifiers systems which differ in a number of subtle technical ways. There is no one rule for choosing among them according to a given problem. This chapter analyzes the major kinds of classifiers systems in order to suggest a generic model (called GEMEAU) that is common to all of them. GEMEAU was developed for a number of different applications and it is applied to timetables problem in order to easily test differents hypotheses.

Chapter 6 detailed the PEGASE project, a tutoring system for virtual reality learning environments. The context of this research is the creation of human learning environments using virtual reality. We propose the integration of a generic and adaptable intelligent tutoring system (PEGASE) into a virtual environment. The aim of this environment is to instruct the learner, and to assist the instructor. The proposed system is created using a multi-agent system. This system emits a set of knowledge (actions carried out by the learner, knowledge about the field, etc.) which PEGASE uses to make informed decisions. Our study focuses on the representation of knowledge about the environment, and on the adaptable pedagogical agent providing instructive assistance.

7.2   Discussion

Research presented in this document provide a set of partial answers to the problems of entities' adaptive behaviors in participatory virtual environments. We proposed an original approach based on three behavioral properties: the entity learns by doing, it uses an imaginary world of simulation within the simulation to anticipate, and it pays particular attention to human behaviors to guide the adaptation of its own behavior.

Let us return to our primary goal: *artificial intelligence* is "seeking ways to equip computer systems with intellectual abilities comparable to those of human beings." It is clear that current techniques are far from allowing us to compare a virtual entity to a human. Why? Primarily due to the nature of computer programs themselves: determinism, rationality and without free will. Our future research should thus avoid conventional approaches (the concept of freedom *vs* determinism, intuition *vs* reason, creativity *vs* linearity). We will design systems whose behavior would be interpreted as the artificial production of views, opinions, assessments, impressions, desires.

To do so, we can look at research on "dynamic systems". The Di Paolo team, for instance, works on an "artificial free mind" [Di Paolo and Iizuka, 2007]. To go further, work has started following the idea of "artificial conscience", firstly introduced by [Cardon, 1999]. The entities should be able, in the same way as humans, to experience sensations, feelings and ultimately make decisions that we could call free. In this movement, the SENSOPAC project provides the development of the first artificial cerebellum (the cerebellum is the part of the brain that controls motor functions). The project is to implement this man-made cerebellum in a robot to acheive movement and a more natural interaction with living things. The ICEA project, meanwhile, focuses on the anchor of the body in the cognition. The project's main goal was to develop an artificial system which integrates cognition, emotion and self-maintenance, inspired by the architecture and physiology of the mammalian brain. This innovative research led us to ask ourselves which path to take next?

7.3   Outlook

Our future research will consider virtual entities at the same level as human, by integrating human characteristics that are currently lacking in existing artificial intelligence mechanisms (creativity, free will, intuition emerging proposal, action guided by a sense, etc.). The goal is to design computer systems whose behavior would be interpreted as the artificial production of views, opinions, assessments, impressions, desires.

7.3.1   A possibility

To meet these challenges, Pierre De Loor proposes to "automate" virtual entities [De loor et al., 2009]. To go further, the author proposes to rely on figure 7.1 for the design of an artificial intelligence. This figure contains three parts:

1. **Adaptation** proposes to regulate behavior using artificial intelligence techniques;

2. **Evolution** seeks to produce new behaviors inspired by ontogenesis and morphogenesis;

3. **Proposition** aims at the emergence of creative behaviors, assessments, impressions, desires.
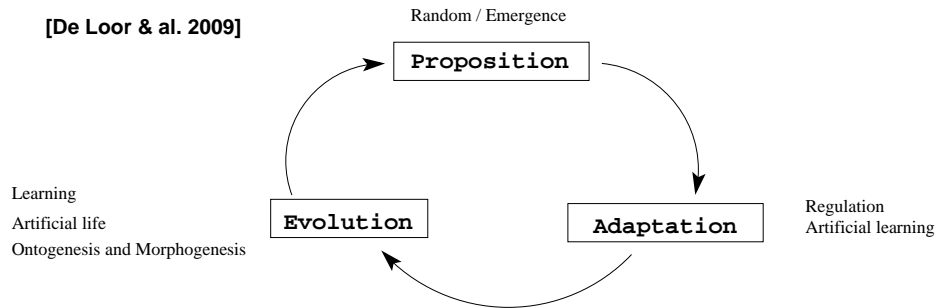
FIGURE 7.1 – Three poles describing the characteristics expected for a virtual entity, according to [De loor et al., 2009]

Behavioral modeling of an autonomous entity would ensure compliance with these three poles. This raises a number of questions:

▷ What models are used to ensure the functions of each pole?

▷ What models are used to manage the interactions between poles?

7.3.2 Ensuring the functions of each pole

We consider that the concepts involved in each pole can be implemented with current technology:

1. **Adaptation**: these function could be ensured by techniques described in this paper, *e.g.* in our work on fuzzy cognitive maps or classifiers systems.

2. **Evolution**: this function could be guaranteed by the techniques described in this paper, particularly in our works on classifiers systems. Note that these techniques are difficult to use online. According to [Manach and De Loor, 2009], we will avoid to use classical genetic algorithm.

3. **Proposition**: the goal is to introduce "artificial creativity". A first proposal was made by using recurrent neural networks [Simaõ, 2008,Manach and De Loor, 2009]. However, these studies are exploratory and are currently applied to minimalist applications.

We propose to direct our future work towards the use of adaptive and proactive entities, performing the processes involved in each of the poles. Each pole would be a multi-agent system (MAS) with the objective to propose to adapt or change the entity's behavior (see
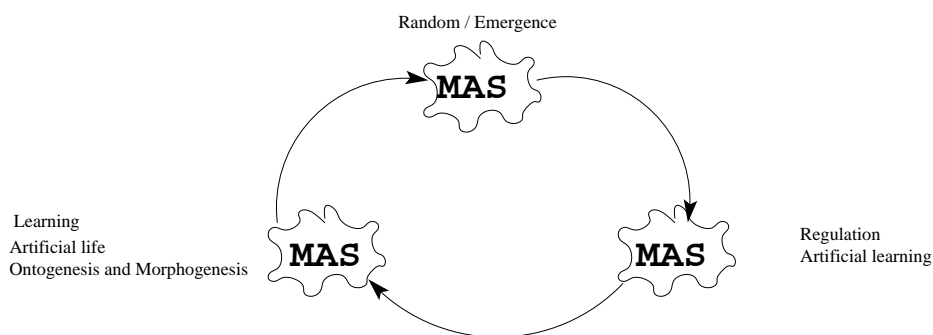
FIGURE 7.2 – Three multi-agent systems to model the behavior of an entity

figure 7.2). The behavioral model of each agent can be simple (recurrent network, probabilistic model, classifier system etc.). Each MAS will be held to cooperate.

Why MAS ? MAS have several important characteristics that can be used to solve problems that are difficult or impossible for a monolithic system to solve (autonomy, local views and decentralization), wich is consistent to the idea of different poles.

7.3.3 Managing interactions between poles

An additional difficulty is to maintain the coherence of the overall behavior from the three poles. The three MAS will thus interact and compete to ensure their own function for each pole. To simulate the overall evolution, interactions between the 3 MAS need to be organized.

7.3.4 Assessment

These perspectives raise several questions which must be answered:

> ▷ What models are needed to achieve competition between the three MAS?

> ▷ What models are required to achieve cooperation between each MAS?

> ▷ What models can be introduced to ensure the pole "Proposition"?

Future research will keep in mind the three properties that we have emphasized along this paper: learning by doing, the principle of simulation within the simulation and the human as a model.

We will also work in cooperation with people working to put humans and virtual entities on the same level, and we will monitor more specifically works aiming at immersing users in virtual environments. The idea is to "renounce our flesh" in a virtual environment, providing an universe free from all physical and biological constraints, where the differences between all forms of intelligence would be abolished.

Personal publications

This section enumerates the papers written during these research.

——————————— **International journals** ———————————

[1] C. Buche, A. Jeannin-Girardon, and P. De Loor. Simulation theory and anticipation as a basis for interactive virtual character in an uncertain world. application to a human-virtual characters interaction for juggling. *Computer Animation and Virtual Worlds (CAVW), Computer Animation and Social Agents (CASA'11) Special Issue*, 22(2-3):133–139, 2011.

[2] C. Buche and R. Querrec. An expert system manipulating knowledge to help human learners into virtual environment. *Expert Systems With Applications (ESWA)*, 38(7):8446–8457, 2011.

[3] C. Buche, C. Bossard, R. Querrec, and P. Chevaillier. PEGASE: A generic and adaptable intelligent system for virtual reality learning environments. *International Journal of Virtual Reality (IJVR)*, 9(2):73–85, 2010.

[4] C. Buche, P. Chevaillier, A. Nédélec, M. Parenthoën, and J. Tisseau. Fuzzy cognitive maps for the simulation of individual adaptive behaviors. *Computer Animation and Virtual Worlds (CAVW)*, 21(6):573–587, 2010.

[5] T. Thanh Hai, C. Buche, R. Querrec, and J. Tisseau. Modeling of errors realized by a human learner in virtual environment for training. *International Journal of Computers, Communications & Control (IJCCC)*, IV(1):73–81, March 2009.

[6] C. Bossard, G. Kermarrec, C. Buche, and J. Tisseau. Transfer of training in virtual environments: A new challenge. *Virtual Reality*, (12):151–161, 2008.

[7] C. Buche, R. Querrec, P. Chevaillier, and G. Kermarrec. Apports des systèmes tutoriaux intelligents et de la réalité virtuelle à l'apprentissage de compétences. *In Cognito – Cahiers Romans de Sciences Cognitives (CRSC)*, 2(2):51–83, 2006.

[8] C. Buche, R. Querrec, P. De Loor, and P. Chevaillier. MASCARET : A pedagogical multi-agent system for virtual environment for training. *International Journal of Distance Education Technologies (JDET)*, 2(4):41–61, 2004.

[9] R. Querrec, C. Buche, E. Maffre, and P. Chevaillier. Multiagents systems for virtual environment for training. application to fire-fighting. *International Journal of Computers and Applications (IJCA)*, 1(1):25–34, 2004.

—————— **National journals** ——————

[10] D. Pasco, C. Bossard, C. Buche, and G. Kermarrec. Utiliser les jeux vidéos actifs pour promouvoir l'activité physique (une revue de littŕature). *Sport Science Review*, XIX(5-6):77–93, 2010.

[11] C. Buche, R. Querrec, P. De Loor, P. Chevaillier, and J. Tisseau. PEGASE: un système tutoriel intelligent générique et adaptatif en environnement virtuel. *Revue des Sciences et Technologies de l'Information, série Techniques et Sciences Informatiques (RSTI–TSI)*, 28(8):1051–1076, 2009.

[12] C. Buche, C. Septseault, and P. De Loor. Les systèmes de classeurs. une présentation générale. *Revue des Sciences et Technologies de l'Information, série Techniques et Sciences Informatiques (RSTI–TSI)*, 25:963–990, 2006.

[13] C. Buche, C. Septseault, and P. De Loor. Proposition d'un modèle générique pour l'implémentation d'une famille de systèmes de classeurs. *Revue des Sciences et Technologies de l'Information, série Intelligence Artificielle (RSTI–RIA)*, 20(1):63–88, 2006.

[14] J. Tisseau, M. Parenthoën, C. Buche, and P. Reignier. Comportements perceptifs d'acteurs virtuels autonomes. une application aux cartes cognitives floues. *Revue des Sciences et Technologies de l'Information, série Techniques et Sciences Informatiques (RSTI–TSI)*, 24(10):1259–1293, 2005.

—————— **International conferences** ——————

[15] R. Querrec, C. Buche, F. Lecorre, and F. Harrouet. Agent metamodel for virtual reality applications. In Dominik Ryzko, Henryk Rybinski, Piotr Gawrysiak, and Marzena Kryszkiewicz, editors, *19th International Symposium on Methodologies for Intelligent Systems (ISMIS'11)*, volume 369, pages 81–90. Springer Berlin / Heidelberg, 2011.

[16] C. Buche and P. De Loor. Generic model for experimenting and using a family of classifiers systems: description and basic applications. In L. Rutkowski, editor, *International Conference on Artificial Intelligence and Soft Computing (ICAISC'10), Part I, LNCS/LNAI 6113*, pages 299–306. Springer, 2010.

[17] F. Tence, C. Buche, P. De Loor, and O. Marc. The challenge of believability in video games: Definitions, agents models and imitation learning. In Wenji Mao and Lode Vermeersch, editors, *2nd Asian Conference on Simulation and AI in Computer Games (GAMEON-ASIA'10)*, pages 38–45. Eurosis, 2010.

[18] F. Tence, C. Buche, P. De Loor, and O. Marc. Learning a representation of a believable virtual character's environment with an imitation algorithm. In M. Al-Akaidi, editor, *First annual Pan-Arabic International Conference on Intelligent Games and Simulation (GAMEON-ARABIA'10)*, pages 141–145. Eurosis, 2010.

[19] F. Tence and C. Buche. Automatable evaluation method oriented toward behaviour believability for video games. In *International Conference on Intelligent Games and Simulation (GAMEON'08)*, pages 39–43, 2008.

[20] T. Thanh Hai, C. Buche, and J. Tisseau. Modeling of errors realized by a learner in virtual environment for training. In *International Conference on Virtual Learning (ICVL'08)*, pages 71–80, 2008.

[21] C. Buche and R. Querrec. Intelligent tutoring system for MASCARET. In Simon Richir and Bernard Taravel, editors, 7th *virtual reality international conference (VRIC'05)*, pages 105–108, April 2005.

[22] C. Buche, R. Querrec, and P. De Loor. Système tutoriel intelligent pour l'apprentissage de travail procedural et collaboratif. In A. Herzig, Y. Lespérance, and A.I. Mouaddib, editors, *Troisièmes journées francophones Modèle Formels de l'Interaction (MFI'05)*, pages 205–210, 2005.

[23] C. Buche, R. Querrec, and C. Le Gall. Intelligent tutoring system for procedural and collaborative training. In 8th *World Conference on Computers in Education (WCCE'05)*, University of Stellenbosch, Cape Town, South Africa, 2005.

[24] C. Buche and R. Querrec. Simulate pedagogical reasoning in a virtual environment for training. In *International Conference on Computers and Advanced Technology for Education (CATE'05)*, pages 183–187, Aruba, 2005.

[25] D. M. Popovici, C. Buche, R. Querrec, and F. Harrouet. An interactive agent-based learning environment for children. In M. Nakajima, Y. Hatori, and A. Sourin, editors, *International Conference on Cyberworlds (CW'04)*, pages 233–240, Tokyo, Japan, 2004. IEEE Computer Society.

[26] C. Buche, R. Querrec, P. De Loor, and P. Chevaillier. MASCARET : Pedagogical multi-agents system for virtual environment for training. In T.L. Kunii, S.H. Soon, and A. Sourin, editors, *International Conference on Cyberworlds (CW'03)*, pages 423–430, Singapore, December 2003. School of Computer Engeneering, Nanyang Technological Unviversity, IEEE Computer Society.

[27] C. Buche, R. Querrec, E. Maffre, P. Chevaillier, and P. De Loor. MASCARET: multi-agent system for virtual environment for training. In Simon Richir, Paul Richard, and Bernard Taravel, editors, 5th *virtual reality international conference (VRIC'03)*, pages 159–164, Laval, France, 2003.

[28] R. Querrec, C. Buche, E. Maffre, and P. Chevaillier. Multiagents systems for virtual environnement for training. In Vladimir Uskov, editor, *International Conference on Computers and Advanced Technology in Education (CATE'03)*, pages 647–652, Rhodes, Greece, 2003. ACTA Press.

[29] R. Querrec, C. Buche, E. Maffre, and P. Chevaillier. SécuRéVi : virtual environments for fire-fighting training. In Simon Richir, Paul Richard, and Bernard Taravel, editors, 5th virtual reality international conference (VRIC'03), pages 169–175, Laval, France, 2003.

[30] C. Buche, M. Parenthoën, and J. Tisseau. Learning by imitation of behaviors for autonomous agents. In Q. Medhi, N. Gough, and M. Cavazza, editors, 3rd International Conference on Intelligent Games and Simulation (GAME-ON'02), pages 89–93, London, United Kingdom, 2002. SCS.

[31] M. Parenthoën, C. Buche, and J. Tisseau. Action learning for autonomous virtual actors. In R. V. Mayorga and A. Segovia-De Los Ríos, editors, 3rd International Symposium on Robotics and Automation (ISRA'02), pages 549–554, Toluca, Mexico, 2002.

Bibliography

[Amokrane et al., 2008] Amokrane, K., Lourdeaux, D., and Burkhardt, J. (2008). Hera: Learner tracking in a virtual environment. *IJVR : International Journal of Virtual Reality*, 7(3):23–30.

[Axelrod, 1976] Axelrod, R. (1976). *Structure of decision*. Princeton University Press, USA.

[Bates, 1992a] Bates, J. (1992a). The nature of characters in interactive worlds and the Oz project. Technical Report CMU-CS-92-200, School of Computer Science, Carnegie Mellon University.

[Bauckhage et al., 2007] Bauckhage, C., Gorman, B., Thurau, C., and Humphrys, M. (2007). Learning Human Behavior from Analyzing Activities in Virtual Environments. *MMI-Interaktiv*, 12:3–17.

[Bergson, 1896] Bergson, H. (1896). *Matière et mémoire*.

[Berthoz, 1997] Berthoz, A. (1997). *Le sens du mouvement*. Odile Jacob, Paris.

[Berthoz, 2003] Berthoz, A. (2003). *La décision*. Odile Jacob.

[Bossard et al., 2009] Bossard, C., Benard, R., De Loor, P., Kermarrec, G., and Tisseau, J. (2009). An exploratory evaluation of virtual football player's believability. In *In Richir, S. & Shirai, A. (Eds). Proceedings of 11th Virtual Reality International Conference (VRIC'09)*.

[Brooks, 1990] Brooks, R. (1990). Elephants don't play chess. *Robotics and Autonomous Systems*, 6(1&2):3–15.

[Brunia, 1999] Brunia, C. (1999). Neural aspects of anticipatory behavior. *Acta Psychologica*, 101:213–242.

[Bryant and Miikkulainen, 2006] Bryant, B. and Miikkulainen, R. (2006). Evolving stochastic controller networks for intelligent game agents. In *Proceedings of the 2006 Congress on Evolutionary Computation (CEC 2006)*, pages 3752–3759.

[Buche et al., 2010a] Buche, C., Bossard, C., Querrec, R., and Chevaillier, P. (2010a). PE-GASE: A generic and adaptable intelligent system for virtual reality learning environments. *International Journal of Virtual Reality (IJVR)*, 9(2):73–85.

[Buche et al., 2010b] Buche, C., Chevaillier, P., Nédélec, A., Parenthoën, M., and Tisseau, J. (2010b). Fuzzy cognitive maps for the simulation of individual adaptive behaviors. *Computer Animation and Virtual Worlds (CAVW)*. [WoS].

[Buche and De Loor, 2010] Buche, C. and De Loor, P. (2010). Generic model for experimenting and using a family of classifiers systems: description and basic applications. In Rutkowski, L., editor, *International Conference on Artificial Intelligence and Soft Computing (ICAISC'10), Part I, LNCS/LNAI 6113*, pages 299–306. Springer.

[Buche et al., 2011] Buche, C., Jeannin-Girardon, A., and De Loor, P. (2011). Simulation theory and anticipation as a basis for interactive virtual character in an uncertain world. application to a human-virtual characters interaction for juggling. *Computer Animation and Virtual Worlds (CAVW), Computer Animation and Social Agents (CASA'11) Special Issue*, 22(2-3):133–139.

[Buche et al., 2002] Buche, C., Parenthoën, M., and Tisseau, J. (2002). Learning by imitation of behaviors for autonomous agents. In Medhi, Q., Gough, N., and Cavazza, M., editors, 3rd *International Conference on Intelligent Games and Simulation (GAME-ON'02)*, pages 89–93, London, United Kingdom. SCS.

[Buche and Querrec, 2011] Buche, C. and Querrec, R. (2011). An expert system manipulating knowledge to help human learners into virtual environment. *Expert Systems With Applications (ESWA)*, 38(7):8446–8457.

[Buche et al., 2006c] Buche, C., Septseault, C., and De Loor, P. (2006c). Les systèmes de classeurs. une présentation générale. *Revue des Sciences et Technologies de l'Information, série Techniques et Sciences Informatiques (RSTI–TSI)*, 25:963–990.

[Buche et al., 2006d] Buche, C., Septseault, C., and De Loor, P. (2006d). Proposition d'un modèle générique pour l'implémentation d'une famille de systèmes de classeurs. *Revue des Sciences et Technologies de l'Information, série Intelligence Artificielle (RSTI–RIA)*, 20(1):63–88.

[Burkert et al., 2007] Burkert, O., Kadlec, R., Gemrot, J., Bida, M., Havlicek, J., Dorfler, M., and Brom, C. (2007). Towards Fast Prototyping of IVAs Behavior: Pogamut 2. In *Intelligent Virtual Agents*, volume 4722, pages 362–363. Springer.

[Butz and Pelikan, 2001] Butz, M. and Pelikan, M. (2001). Analyzing the evolutionary pressures in xcs. In Spector, L., Goodman, E. D., Wu, A., Langdon, W., Voigt, H.-M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M. H., and Burke, E., editors, *Genetic and Evolutionary Computation Conference (GECCO'01)*, pages 935–942, San Francisco, California, USA. Morgan Kaufmann.

[Butz and Wilson, 2002] Butz, M. and Wilson, W. (2002). An algorithmic description of xcs. *Journal of Soft Computing*, 6(3–4):144–153.

[Butz et al., 2003b] Butz, M. V., Sigaud, O., and Gérard, P. (2003b). Anticipatory behavior: Exploiting knowledge about the future to improve current behavior. In Butz, M. V., Sigaud, O., and Gérard, P., editors, *LNCS 2684 : Anticipatory Behavior in Adaptive Learning Systems*, pages 1–12. Springer-Verlag.

[Butz et al., 2007] Butz, M. V., Sigaud, O., Pezzulo, G., and Baldassarre, G. (2007). Brains, anticipations, individual and social behavior: an introduction to anticipatory behavior systems. In Butz, M. V., Sigaud, O., Pezzulo, G., and Baldassarre, G., editors, *LNCS 4520 : Anticipatory Behavior in Adaptive Learning Systems: From Brains to Individual and Social Behavior*, pages 1–18. Springer-Verlag.

[Calais, 2008] Calais, G. (2008). Fuzzy cognitive maps theory: Implications for interdisciplinary reading: National implications. *FOCUS On Colleges, Universities, and Schools*, 2.

[Cardon, 1999] Cardon, A. (1999). *Conscience artificielle et systèmes adaptatifs*. Eyrolles, France.

[Carver and Lesser, 1992] Carver, M. and Lesser, V. (1992). The evolution of blackboard control architectures. Technical Report UM-CS-1992-071, Department of Computer Science, University Massachusetts.

[Cass, 2002] Cass, S. (2002). Mind games. *IEEE Spectrum*, 39(12):40–44.

[Cavazza et al., 2001] Cavazza, M., Charles, F., and Mead, S. (2001). Characters in search of an author : Ai-based virtual staorytelling. In *Virtual Storytelling, International Conference ICVS 2001*, number 2197 in LNCS, pages 145–154. Springer.

[Cavazza et al., 2003] Cavazza, M., Charles, F., and Mead, S. (2003). Interactive storytelling: from AI experiment to new media. In *ICEC '03: Proceedings of the second international conference on Entertainment computing*, pages 1–8, Pittsburgh, PA, USA. Carnegie Mellon University.

[Chaib-Draa, 2002] Chaib-Draa, B. (2002). Causal maps: theory, implementation, and practical applications in multiagent environments. *IEEE Transactions on Knowledge and Data Engineering*, 14(6):1201–1217.

[Chevaillier et al., 2000] Chevaillier, P., Harrouet, F., Reignier, P., and Tisseau, J. (2000). Virtual Reality and Multi-agent Systems for Manufacturing System Interactive Prototyping. *International Journal of Design and Innovation Research*.

[Cliff and Ross, 1995] Cliff, D. and Ross, S. (1995). Adding Temporary Memory to ZCS. Technical Report CSRP347, School of Cognitive and Computing Sciences, University of Sussex.

[Colwill and Rescorla, 1985] Colwill, R. and Rescorla, R. (1985). Postconditioning devaluation of a reinforcer affects instrumental responding. *Journal of experimental psychology. Animal behavior processes*, 11(1):120–132.

[Cornuéjols et al., 2002] Cornuéjols, A., Miclet, L., and Kodratoff, Y. (2002). *Apprentissage artificiel: concepts et algorithmes*. Eyrolles, Paris.

[Cybenko, 1989] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2:303–314.

[Davidsson, 2003a] Davidsson, P. (2003a). A Framework for Preventive State Anticipation. *Anticipatory Behavior in Adaptive Learning Systems: Foundations, Theories, and Systems*.

[De Loor and Chevaillier, 2003] De Loor, P. and Chevaillier, P. (2003). Solving distributed and dynamic constraints using an emotional metaphor: Application to the timetabling problem. In *Euro-Inform - Istanbul*.

[De loor et al., 2009] De loor, P., Manach, K., and Tisseau, J. (2009). Enaction-based artificial intelligence: Toward co-evolution with humans in the loop. *Minds and Machines*, 19(3):319–343.

[Del Bimbo and Vicario, 1995] Del Bimbo, A. and Vicario, E. (1995). Specification by example of virtual agents behavior. *IEEE Transactions on Visualization and Computer Graphics*, 1(4):350–360.

[Dempster et al., 1977] Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.

[Di Paolo and Iizuka, 2007] Di Paolo, E. and Iizuka, H. (2007). How (not) to model autonomous behaviour.

[Dickerson and Kosko, 1994] Dickerson, J. and Kosko, B. (1994). Virtual Worlds as Fuzzy Cognitive Maps. *Presence*, 3(2):173–189.

[Dorigo, 1995] Dorigo, M. (1995). Alecsys and the AutonoMouse: Learning to Control a Real Robot by Distributed Classifier Systems. *Machine Learning*, 19:209–240.

[Dorigo and Bersini, 1994] Dorigo, M. and Bersini, H. (1994). A Comparison of Q-Learning and Classifier Systems. In *From Animals to Animats 3: Third International Conference on Simulation of Adaptive Behavior (SAB94)*, pages 248–255.

[Dufossé et al., 1985] Dufossé, M., Hugon, M., and Massion, J. (1985). Postural forearm changes induced by predictable in time or voluntary triggered unloading in man. *Brain research*, 60:330–334.

[Edward et al., 2008] Edward, L., Lourdeaux, D., Lenne, D., Barthes, J., and Burkhardt, J. (2008). Modelling autonomous virtual agent behaviours in a virtual environment for risk. *IJVR : International Journal of Virtual Reality*, 7(3):13–22.

[El-Kechaï and Després, 2006] El-Kechaï, N. and Després, C. (2006). A plan recognition process, based on a task model, for detecting learner's erroneous actions. In *Intelligent Tutoring Systems*, pages 329–338.

[Epicure, 200] Epicure (-200). *Lettre à Hérodote*.

[Florez-Larrahondo, 2005] Florez-Larrahondo, G. (2005). *Incremental learning of discrete hidden Markov models*. PhD thesis, Mississippi State University.

[Fritzke, 1995] Fritzke, B. (1995). A growing neural gas network learns topologies. In *Advances in Neural Information Processing Systems 7*, pages 625–632. MIT Press.

[Fuchs et al., 2001a] Fuchs, P., Moreau, G., and Papin, J. (2001a). *Le traité de la réalité virtuelle*. Presses de l'Ecole des Mines, Paris.

[Gallese, 2000] Gallese, V. (2000). The inner sense of action: agency and motor representations. *Journal of Consciousness Studies*, 7(10):23–40.

[Gaussier et al., 1998] Gaussier, P., Moga, S., Quoy, M., and Banquet, J. (1998). From perception-action loops to imitation process: a bottom-up approach of learning by imitation. *Applied Artificial Intelligence*, 12(7):701–727.

[Gérard et al., 2002] Gérard, P., Stolzmann, W., and Sigaud, O. (2002). Yacs : a new learning classifier system using anticipation. *Journal of Soft Computing : Special Issue on Learning Classifier Systems*.

[Gibson, 1979] Gibson, J. (1979). *The ecological approach to visual perception*. Lawrence Erlbaum Associates, London.

[Goldberg, 1989] Goldberg, D. (1989). *Search, Optimization and Machine Learning*, chapter Genetic Algorithms. Reading MA Addison Wesley.

[Gorman and Humphrys, 2007] Gorman, B. and Humphrys, M. (2007). Imitative learning of combat behaviours in first-person computer games. In *Proceedings of CGAMES 2007, the 11th International Conference on Computer Games: AI, Animation, Mobile, Educational & Serious Games*.

[Gorman et al., 2006b] Gorman, B., Thurau, C., Bauckhage, C., and Humphrys, M. (2006b). Believability testing and bayesian imitation in interactive computer games. In *From Animals to Animats 9*, volume 4095, pages 655–666. Springer.

[Groom et al., 2009b] Groom, V., Nass, C., Chen, T., Nielsen, A., Scarborough, J. K., and Robles, E. (2009b). Evaluating the effects of behavioral realism in embodied agents. *International Journal of Human-Computer Studies*, 67(10):842–849.

[Hafner, 2000] Hafner, V. (2000). Learning places in newly explored environments. In *in Meyer, Berthoz, Floreano, Roitblat and Wilson (Eds.), SAB2000 Proceedings Supplement Book, Publication of the International Society for Adaptive Behavior*.

[Hagiwara, 1992] Hagiwara, M. (1992). Extended fuzzy cognitive maps. In *IEEE International Conference on Fuzzy Systems*.

[Hayes and Ford, 1995] Hayes, P. and Ford, K. (1995). Turing test considered harmful. In *International Joint Conference on Artificial Intelligence*, volume 14, pages 972–977. Lawrence Erlbaum Associates LTD.

[Hebb, 1949] Hebb, D. (1949). *The Organization of Behaviour*. John Wiley and Sons, New York, USA.

[Heguy et al., 2002] Heguy, O., Sanza, C., Berro, A., and Duthen, Y. (2002). GXCS: A generic classifier system and its application in a real time cooperative behavior simulations. In *International Symposium and School on Advanced Distribued System (ISADS'02)*.

[Held and Durlach, 1991] Held, R. and Durlach, N. (1991). Telepresence, time delay and adaptation. *Pictorial communication in virtual and real environments*, pages 232–246.

[Herviou and Maisel, 2006] Herviou, D. and Maisel, E. (2006). ARéViRoad : a virtual reality tool for traffic simulation. In *Proceedings of Urban Transport*, pages 297–306.

[Hesslow, 2002] Hesslow, G. (2002). Conscious thought as simulation of behaviour and perception. *Trends in Cognitive Sciences*, 6(6):242–247.

[Ho et al., 2008] Ho, C.-C., MacDorman, K. F., and Pramono, Z. A. D. D. (2008). Human emotion and the uncanny valley: a glm, mds, and isomap analysis of robot video ratings. In *HRI '08: Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*, pages 169–176, New York, NY, USA. ACM.

[Hoffmann, 2003] Hoffmann, J. (2003). Anticipatory Behavioral Control. *Anticipatory Behavior in Adaptive Learning Systems: Foundations, Theories, and Systems*.

[Hoffmann et al., 2007] Hoffmann, J., Berner, M., Butz, M., Herbort, O., Kiesel, A., Kunde, W., and Lenhard, A. (2007). Explorations of anticipatory behavioral control (abc): A report from the cognitive psychology unit of the university of waburg. In *Cognitive Processing*, pages 133–142.

[Holland and Reitman, 1978] Holland, J. and Reitman, J. (1978). Cognitive systems based on adaptive algorithms. In Waterman, D. A. and Hayes-Roth, F., editors, *Pattern-directed inference systems*. New York: Academic Press.

[Holland, 1975b] Holland, J. H. (1975b). *Adaptation in Natural and Artifcial Systems*. university of Michigan Press.

[Holland, 1985] Holland, J. H. (1985). Properties of the bucket brigade algorithm. In *Proc. of the International Conference on Genetic Algorithms and Their Applications*, pages 1–7, Pittsburgh, PA.

[Holland, 1986] Holland, J. H. (1986). Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., editors, *Machine Learning: An Artificial Intelligence Approach*, volume 2, pages 593–623. Kaufmann, Los Altos, CA.

[Hubal, 2008] Hubal, R. (2008). Embodied tutors for interaction skills simulation training. *IJVR : International Journal of Virtual Reality*, 7(1):1–8.

[Hugon et al., 1982] Hugon, M., Massion, J., and Wiesendanger, M. (1982). Anticipatory postural changes induced by active unloading and comparison with passive unloading in man. *Pflugers Arch*, 393:292–296.

[Hume, 1740] Hume, D. (1740). *Traité de la nature humaine*.

[Husserl, 1992] Husserl, E. (1992). *Méditations cartésiennes*. Vrin.

[Isla, 2001] Isla, D. (2001). *The virtual hippocampus : spatial common sense for synthetic creatures*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology.

[Isla, 2005] Isla, D. (2005). Handling complexity in the halo 2 ai. *Game Developers Conference*, page 12.

[Johansson and Balkenius, 2007] Johansson, B. and Balkenius, C. (2007). An Experimental Study of Anticipation in Simple Robot Navigation. *Anticipatory Behavior in Adaptive Learning Systems: From Brains to Individual and Social Behavior, Springer, Heidelberg*.

[Julliard and Gibet, 1999] Julliard, F. and Gibet, S. (1999). Reactiva'motion project: Motion synthesis based on a reactive representation. In *GW '99: Proceedings of the International Gesture Workshop on Gesture-Based Communication in Human-Computer Interaction*, pages 265–268, London, UK. Springer-Verlag.

[Kallmann and Thalmann, 1998] Kallmann, M. and Thalmann, D. (1998). Modeling objects for interaction tasks. In *Proceedings of Computer Animation and Simulation'98*, pages 73–86.

[Klesen et al., 2000] Klesen, M., Szatkowski, J., and Lehmann, N. (2000). The Black Sheep: Interactive Improvisation in a 3D Virtual World. In *I3*, pages 77–80.

[Kosko, 1986b] Kosko, B. (1986b). Fuzzy Cognitive Maps. *International Journal of Man-Machine Studies*, 24:65–75.

[Kosko, 1988] Kosko, B. (1988). Hidden Patterns in Combined and Adaptative Knowledge Networks. *International Journal of Approximate Reasoning*, 2:337–393.

[Kosko, 1992] Kosko, B. (1992). *Neural networks and fuzzy systems: a dynamical systems approach to machine intelligence*. Engelwood Cliffs, Prentice-Hall edition.

[Koulouriotis et al., 2003] Koulouriotis, D., Diakoulakis, I., Emiris, D. M., Antonidakis, E., and Kaliakatsos, I. (2003). Efficiently Modeling and Controlling Complex Dynamic Systems using Evolutionary Fuzzy Cognitive Maps. *International Journal of Computational Cognition*, 1(2):41–65.

[Kunde et al., 2002] Kunde, W., Hoffmann, J., and Zellmann, P. (2002). The impact of anticipated action effects on action planning. *Acta Psychologica*, 109(2):137–155.

[Kunde et al., 2004] Kunde, W., Koch, I., and Hoffmann, J. (2004). Anticipated action effects affect the selection, initiation, and execution of actions. *The Quarterly Journal of Experimental Psychology Section A*, 57(1):87–106.

Labbé, V. and Sigaud, O. (2004a). Anticipation of periodic movements in real time 3d environments. In *Anticipatory Behavior in Adaptive Learning Systems*.

[Labbé and Sigaud, 2004b] Labbé, V. and Sigaud, O. (2004b). Anticipation of Periodic Movements in Real Time 3D Environments. In *Proceedings of the ABiALS 2004 Workshop*, Los Angeles, CA.

[Lahlou, 2007] Lahlou, S. (2007). Human activity modeling for systems design: A transdisciplinary and empirical approach. pages 512–521.

[Laird and Duchi, 2000] Laird, J. and Duchi, J. (2000). Creating human-like synthetic characters with multiple skill levels: a case study using the Soar Quakebot. In *Simulating Human Agents, Papers from the 2000 AAAI Fall Symposium*, pages 75–79.

[Laird, 2001c] Laird, J. E. (2001c). It knows what you're going to do: adding anticipation to a quakebot. In Müller, J. P., Andre, E., Sen, S., and Frasson, C., editors, *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 385–392, Montreal, Canada. ACM Press.

[Lanzi and Colombetti, 1999] Lanzi, P. and Colombetti, M. (1999). An extension to the XCS classifier system for stochastic environments. In *Genetic and Evolutionary Computation Conference*, volume 1, pages 353–360, Orlando, Florida, USA. Morgan Kaufmann.

[Lanzi and Wilson, 1999] Lanzi, P. and Wilson, S. (1999). Optimal classifier system performance in non-Markov environments. Technical Report 99.36, Département d'Electronique et d'Information - Polytechnique de Milan.

[Laroque et al., 2010] Laroque, P., Gaussier, N., Cuperlier, N., Quoy, M., and Gaussier, P. (2010). Cognitive map plasticity and imitation strategies to improve individual and social behaviors of autonomous agents. *Journal of Behavioral Robotics*.

[Le Hy, 2007] Le Hy, R. (2007). *Programmation et apprentissage bayésien de comportements pour des personnages synthétiques, application aux personnages de jeux vidéos*. PhD thesis, Institut national polytechnique de Grenoble.

[Le Hy et al., 2004] Le Hy, R., Arrigoni, A., Bessiere, P., and Lebeltel, O. (2004). Teaching bayesian behaviours to video game characters. *Robotics and Autonomous Systems*, 47:177–185.

[Levesque, 2003] Levesque, P. (2003). Creation and use of 3d as-built models at EDF. In *FIG Working Week 2003*.

[Lewin, 1936] Lewin, K. (1936). *Principles of topological psychology*. McGraw-Hill, New York.

[Livingstone, 2006] Livingstone, D. (2006). Turing's test and believable AI in games. *Computers in Entertainment*, 4(1):6.

[Lourdeaux et al., 2002] Lourdeaux, D., Burkhardt, J., Bernard, F., and Fuchs, P. (2002). Relevance of an intelligent agent for virtual reality training. *International Journal of Continuous Engineering and Life-long Learning*, 12(1/2/3/4):131–143.

[Loyall, 1997b] Loyall, A. B. (1997b). *Believable agents: building interactive personalities*. PhD thesis, Carnegie Mellon University.

[Mac Namee, 2004] Mac Namee, B. (2004). *Proactive persistent agents: using situational intelligence to create support characters in character-centric computer games*. PhD thesis, University of Dublin.

[Mac Namee, 2004] Mac Namee, B. (2004). *Proactive Persistent Agents: Using Situational Intelligence to Create Support characters in Character-Centric Computer Games*. PhD thesis, Trinity College Dublin.

[Maes, 1989] Maes, P. (1989). The dynamics of action selection. In *Proceedings of the international Joint Conference on Artificial Intelligence, IJCAI'89*.

[Manach and De Loor, 2009] Manach, K. and De Loor, P. (2009). Guiding for associative learning: How to shape dynamical cognition. In *ECAL: 10th European Conference on Artificial Life. LNCS/LNAI 5777, 5778.*

[Marion et al., 2007] Marion, N., Septseault, C., Boudinot, A., and Querrec, R. (2007). Gaspar : Aviation management on an aircraft carrier using virtual reality. In *Cyberworlds 2007 proceedings*, pages 15–22.

[Mataric, 2002] Mataric, M. (2002). Visuo-motor primitives as a basis for learning by imitation: linking perception to action and biology to robotics. In Dautenhahn, K. and Nehaniv, C., editors, *Imitation in Animals and Artifacts*, pages 392–422. MIT Press.

[Mateas, 1997] Mateas, M. (1997). An Oz-centric review of interactive drama and believable agents. Technical Report CMU-CS-97-156, Carnegie Mellon University, Pittsburgh, PA, USA.

[Mateas, 1999] Mateas, M. (1999). An Oz-centric review of interactive drama and believable agents. *Lecture Notes in Computer Science*, 1600:297–343.

[Matteucci, 1999] Matteucci, M. (1999). Fuzzy learning classifier system: Issues and architecture. Technical Report 99.71, Dipartimento di Elettronica e Informazione - Politecnico di Milano.

[Meltzoff, 1995] Meltzoff, A. (1995). Understanding the intentions of others: re-enactement of intended acts by 18-month-old children. *Developmental Psychology*, 31:838–850.

[Meyer, 1999] Meyer, C. (1999). *S.A.G.A.C.E.: Solution Algorithmique Génétique pour l'Anticipation de Comportements Evolutifs - Application aux jeux àă information complète et imparfaite.* PhD thesis, Université de Paris 06.

[Mozer, 1995] Mozer, M. (1995). A Focused Backpropagation Algorithm for Temporal Pattern Recognition. In Chauvin, Y. and Rumelhart, D., editors, *Backpropagation: Theory, Architechtures and Applications*, pages 137–169. Lawrence Erlbaum Associates.

[Multon et al., 2001] Multon, F., Ménardais, S., and Arnaldi, B. (2001). Human motion coordination: a juggler as an example. *The Visual Computer*, 17(2):91–105.

[Nédélec et al., 2005] Nédélec, A., Follut., D., Septseault, C., and Rozec, G. (2005). Emotions, Personality and Social Interactions Modelling in a Multiagent Environment. *Proceedings of CASA 2005, Hong Kong (China)*, pages 103–108.

[Norman and Shallice, 1986] Norman, D. and Shallice, T. (1986). Attention to action: willed and automatic control of behavior. *Consciousness and Self-regulation*, 4:1–18.

[Papageorgiou and Groumpos, 2005] Papageorgiou, E. I. and Groumpos, P. P. (2005). A new hybrid method using evolutionary algorithms to train fuzzy cognitive maps. *Applied Soft Computing*, 5(4):409–431.

[Papageorgiou et al., 2004] Papageorgiou, E. I., Stylios, C., and Groumpos, P. P. (2004). Active hebbian learning algorithm to train fuzzy cognitive maps. *International Journal of Approximate Reasoning*, 37(3):219–249.

[Papageorgiou et al., 2006] Papageorgiou, E. I., Stylios, C., and Groumpos, P. P. (2006). Unsupervised learning techniques for fine-tuning fuzzy cognitive map causal links. *International Journal of Human-Computer Studies*, 64(8):727–743.

[Parenthoën et al., 2002b] Parenthoën, M., Tisseau, J., and Morineau, T. (2002b). Believable decision for virtual actors. In *IEEE International Conference on Systems, Man and Cybernetics (IEEE-SMC)*, volume 3, page MP2R3, Tunisia.

[Pezzulo et al., 2007b] Pezzulo, G., Hoffmann, J., and Falcone, R. (2007b). Anticipation and anticipatory behavior. *Cognitive Processing*, 8(2):67–70.

[Pinchbeck, 2008] Pinchbeck, D. (2008). Trigens can't swim: intelligence and intentionality in first person game worlds. In *Proceedings of The Philosophy of Computer Games 2008*. University of Potsdam.

[Pomerol and Brézillon, 2001] Pomerol, J. and Brézillon, P. (2001). About some relationships between knowledge and context. In Akman, V., Bouquet, P., Thomason, R., and Young, R. A., editors, *Modeling and Using Context: Third International and Interdisciplinary Conference, Context 2001*, pages 461–464, Berlin. Springer-Verlag.

[Popovici et al., 2004a] Popovici, D. M., Buche, C., Querrec, R., and Harrouet, F. (2004a). An interactive agent-based learning environment for children. In Nakajima, M., Hatori, Y., and Sourin, A., editors, *International Conference on Cyberworlds (CW'04)*, pages 233–240, Tokyo, Japan. IEEE Computer Society.

[Querrec et al., 2011] Querrec, R., Buche, C., Lecorre, F., and Harrouet, F. (2011). Agent metamodel for virtual reality applications. In Ryzko, D., Rybinski, H., Gawrysiak, P., and Kryszkiewicz, M., editors, *19th International Symposium on Methodologies for Intelligent Systems (ISMIS'11)*, volume 369, pages 81–90. Springer Berlin / Heidelberg.

[Querrec et al., 2004] Querrec, R., Buche, C., Maffre, E., and Chevaillier, P. (2004). Multiagents systems for virtual environment for training. application to fire-fighting. *International Journal of Computers and Applications (IJCA)*, 1(1):25–34.

[Reed, 1993] Reed, E. (1993). The intention of the use a specific affordance: A conceptual framework for psychology. In Wozniak, R. and Fisher, K., editors, *Development in context, acting and thinking in specific environment*, pages 45–76. Lawrence Erlbaum Associates, New York.

[Richard, 1990] Richard, J. F. (1990). *Les activités mentales : Comprendre, Raisonner, Trouver des solutions*. Armand Colin, Paris. ISBN 2-200-2187-0.

[Rickel and Johnson, 1999c] Rickel, J. and Johnson, W. L. (1999c). Animated agents for procedural training in virtual reality : Perception, cognition, and motor control. *Applied Artificial Intelligence*, 13.

[Riedl and Young, 2005] Riedl, M. and Young, R. (2005). An objective character believability evaluation procedure for multi-agent story generation systems. In *Intelligent Virtual Agents*, volume 3661, pages 278–291. Springer.

[Riegler, 2001] Riegler, A. (2001). The role of anticipation in cognition. In *Computing anticipatory systems (CASYS) International Conference. AIP Conference Proceedings*, volume 573, pages 534–541. American Institute of Physics.

[Rizzolatti et al., 1996] Rizzolatti, G., Fadiga, L., Gallese, V., and Fogassi, L. (1996). Premotor cortex and the recognition of motor actions. *Cognitive Brain Research*, 3(2):131–141.

[Rodriguez-Repiso et al., 2007] Rodriguez-Repiso, L., Setchi, R., and Salmeron, J. (2007). Modelling IT Projects Success with Fuzzy Cognitive Maps. *Expert Systems with Applications*, 32(2):543–559.

[Rosen, 1985a] Rosen, R. (1985a). *Anticipatory systems*. Pergamon Press.

[Rumelhart and Mcclelland, 1986] Rumelhart, D. and Mcclelland, J. (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations (Parallel Distributed Processing)*. MIT Press.

[Rummery and Niranjan, 1994] Rummery, G. A. and Niranjan, M. (1994). On-line q-learning using connectionist systems. Technical report.

[Salmeron, 2009] Salmeron, J. L. (2009). Augmented fuzzy cognitive maps for modelling lms critical success factors. *Knowledge-Based Systems*, 22(4):275–278.

[Sanza et al., 2001] Sanza, C., Heguy, O., and Duthen, Y. (2001). Evolution and cooperation of virtual entities with classifier systems. In *Eurographic Workshop on Computer Animation and Simulation*.

[Schultz et al., 1996a] Schultz, A., Grefenstette, J., and Adams, W. (1996a). Roboshepherd: Learning a Complex Behavior. Technical Report AIC-96-017, Naval Research Laboratory, Washington D.C., USA.

[Searle, 1980] Searle, J. (1980). Minds, brains, and programs. *The behavioral and brain sciences*, 3:353–382.

[Sigaud and Wilson, 2007b] Sigaud, O. and Wilson, S. W. (2007b). Learning classifier systems: A survey. *Journal of Soft Computing*, 11(11):1065–1078.

[Silverman et al., 2006] Silverman, B. G., Bharathy, G., O'Brien, K., and Cornwell, J. (2006). Human behavior models for agents in simulators and games: part ii: gamebot engineering with pmfserv. *Presence: Teleoper. Virtual Environ.*, 15(2):163–185.

[Simaõ, 2008] Simaõ, J. (2008). Aperiodic (chaotic) behavior in rnn with homeostasis as a source of behavior novelty: Theory and applications. *Recurrent Neural Networks*.

[Simons and Chabris, 1999] Simons, D. and Chabris, C. (1999). Gorillas in our midst: sustained inattentional blindness for dynamic events. *perception*, 28:1059–1074.

[Siraj et al., 2001] Siraj, A., Bridgesk, S., and Vaughn, R. (2001). Fuzzy cognitive maps for decision support in an intelligent intrusion detection system. In *International Fuzzy Systems Association/ North American Fuzzy Information Processing Society (IFSA/NAFIPS) Conference on Soft Computing*.

[Slater et al., 1995] Slater, M., Usoh, M., and Steed, A. (1995). Taking steps: the influence of a walking technique on presence in virtual reality. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 2(3):201–219.

[Smith, 1980b] Smith, S. F. (1980b). *A Learning System Based on Genetic Adaptive Algorithms*. PhD thesis, University of Pittsburgh.

[Steuer, 1992] Steuer, J. (1992). Defining virtual reality: dimensions determining telepresence. *Journal of Communication*, 42(4):73–93.

[Stoffregen et al., 1999a] Stoffregen, T., Gorday, K., Sheng, Y.-Y., and Flynn, S. (1999a). Perceiving affordances for another person's actions. *Journal of Experimental Psychology: Human Perception and Performance*, 25:120–136.

[Stolzmann, 1998] Stolzmann, W. (1998). Anticipatory classifier systems. In *Third Annual Genetic Programming Conference*, pages 658–664. Morgan Kaufmann.

[Stylios et al., 1997b] Stylios, C., Georgopoulos, V., and Groumpos, P. (1997b). The use of fuzzy cognitive maps in modeling systems. In *5th IEEE Mediterranean Conference on Control and Systems, Paphos*. paper 67 (CD-ROM).

[Stylios et al., 2008] Stylios, C., Georgopoulos, V., Malandraki, G., and Chouliara, S. (2008). Fuzzy Cognitive Map Architectures for Medical Decision Support Systems. *Applied Soft Computing*, 8(3):1243–1251.

[Stylios and Groumpos, 2000] Stylios, C. and Groumpos, P. (2000). Fuzzy cognitive map in modeling supervisory control systems. *Journal of Intelligent & Fuzzy Systems*, 8:83–98.

[Sugeno and Nishida, 1985] Sugeno, M. and Nishida, M. (1985). Fuzzy Control of a Model Car. *Fuzzy Sets and Systems*, 16:103–113.

[Sutton, 1984] Sutton, R. S. (1984). *Temporal Credit Assignement in Reinforcement Learning*. PhD thesis, University of Massachusetts.

[Sutton, 1988c] Sutton, R. S. (1988c). Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44.

[Taber, 1991] Taber, R. (1991). Knowledge processing with fuzzy cognitive maps. *Expert Systems with Applications*, 2:83–87.

[Takadama et al., 1999] Takadama, K., Terano, T., and Shimohara, k. (1999). Can multiagents learn in organization? – analyzing organizational learning-oriented classifier system. In *IJCAI'99 Workshop on Agents Learning about, from and other Agents*.

[Tence and Buche, 2008] Tence, F. and Buche, C. (2008). Automatable evaluation method oriented toward behaviour believability for video games. In *International Conference on Intelligent Games and Simulation (GAMEON'08)*, pages 39–43.

[Tence et al., 2010a] Tence, F., Buche, C., De Loor, P., and Marc, O. (2010a). The challenge of believability in video games: Definitions, agents models and imitation learning. In Mao, W. and Vermeersch, L., editors, *2nd Asian Conference on Simulation and AI in Computer Games (GAMEON-ASIA'10)*, pages 38–45. Eurosis.

[Tence et al., 2010b] Tence, F., Buche, C., De Loor, P., and Marc, O. (2010b). Learning a representation of a believable virtual character's environment with an imitation algorithm. In Al-Akaidi, M., editor, *First annual Pan-Arabic International Conference on Intelligent Games and Simulation (GAMEON-ARABIA'10)*, pages 141–145. Eurosis.

[Thanh Hai et al., 2009] Thanh Hai, T., Buche, C., Querrec, R., and Tisseau, J. (2009). Modeling of errors realized by a human learner in virtual environment for training. *International Journal of Computers, Communications & Control (IJCCC)*, IV(1):73–81.

[Thomas and Johnston, 1981] Thomas, F. and Johnston, O. (1981). *Disney animation: the illusion of life*. Abbeville Press.

[Thurau et al., 2004b] Thurau, C., Bauckhage, C., and Sagerer, G. (2004b). Learning human-like movement behavior for computer games. In *Proceedings of the 8th International Conference on the Simulation of Adaptive Behavior (SAB'04)*.

[Thurau et al., 2005] Thurau, C., Paczian, T., and Bauckhage, C. (2005). Is bayesian imitation learning the route to believable gamebots? In *GAMEON-NA'2005*, pages 3–9.

[Tisseau, 2001a] Tisseau, J. (2001a). Réalité virtuelle : autonomie in virtuo. Technical report, Habilitation à Diriger des Recherches - Université de Rennes I.

[Tisseau, 2004b] Tisseau, J. (2004b). Réalité virtuelle et complexité, expérimentation in virtuo des systèmes complexes. *Manifeste scientifique du CERV*.

[Tisseau and Harrouet, 2003a] Tisseau, J. and Harrouet, F. (2003a). *Le traité de la réalité virtuelle*, volume 2, chapter Autonomie des entités virtuelles. Presses de l'Ecole des Mines, Paris.

[Tisseau et al., 2005b] Tisseau, J., Parenthoen, M., Buche, C., and Reignier, P. (2005b). Comportements perceptifs d'acteurs virtuels autonomes. une application aux cartes cognitives floues. *Revue des Sciences et Technologies de l'Information, série Techniques et Sciences Informatiques (RSTI–TSI)*, 24(10):1259–1293.

[Tolman, 1948] Tolman, E. (1948). Cognitive Maps in Rats and Men. *Psychological Review*, 55(4):189–208.

[Tolman, 1959] Tolman, E. (1959). Principles of purposive behavior. *Psychology: A Study of a Science*.

[Tomlinson and Bull, 1999b] Tomlinson, A. and Bull, L. (1999b). A zeroth level corporate classifier system. In P-L Lanzi, W. S. and Wilson, S. W., editors, *Second International Workshop on Learning Classifier Systems*. Springer.

[Turing, 1950a] Turing, A. (1950a). Computing machinery and intelligence. *Mind*, 59(236):433–460.

[Turner, 1993] Turner, R. M. (1993). Context-sensitive reasoning for autonomous agents and cooperative distributed problem solving. In *In Proceedings of the IJCAI Workshop on Using Knowledge in its Context*, pages 141–151.

[Urbanowicz and Moore, 2009] Urbanowicz, R. and Moore, J. (2009). Learning classifier systems: A complete introduction, review, and roadmap. *Journal of Artificial Evolution and Applications*, page 25.

[Van Valen, 1973] Van Valen, L. (1973). A new evolutionary law. *Evolutionary Theory*, 1(1):1–30.

[Van Welbergen et al., 2009] Van Welbergen, H., Basten, B. J. H., Egges, A., Ruttkay, Z., and Overmars, M. H. (2009). Real time character animation: A trade-off between naturalness and control. In Pauly, M. and Greiner, G., editors, *Eurographics - State-of-the-Art-Report*, pages 45–72, Munich. Eurographics Association.

[Vaughan et al., 2000] Vaughan, R., Sumpter, N., Henderson, J., Frost, A., and Cameron, S. (2000). Experiments in Automatic Flock Control. *Journal of Robotics and Autonomous Systems*, 31:109–117.

[Voyles et al., 1997] Voyles, R., Morrow, J., and P., K. (1997). Towards gesture-based programming: Shape from motion primordial learning of sensorimotor primitives. *Robotics and Autonomous Systems*, 22(3-4):361–375.

[Watkins, 1989] Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards*. PhD thesis, Cambridge University, Cambridge, England.

[Wenger, 1987] Wenger, E. (1987). *Artificial Intelligence and Tutoring Systems*. Morgan Kaufmann, Los Altos, California.

[Wenstop, 1976] Wenstop, F. (1976). Deductive Verbal Models of Organizations. *International Journal of Man-Machine Studies*, 8:293–311.

[Wetzel, 2004] Wetzel, B. (2004). Step one: Document the problem. In *Challenges in Game Artificial Intelligence: Papers from the 2004 AAAI Workshop, AAAI Press, Menlo Park, CA*, pages 11–15.

[William and Zipser, 1995] William, R. and Zipser, D. (1995). Gradient-based Learning Algorithms for Recurrent Networks and their Computational Complexity. In Chauvin, Y. and Rumelhart, D., editors, *Backpropagation: Theory, Architechtures and Applications*, pages 433–486. Lawrence Erlbaum Associates.

[Wilson, 1987] Wilson, S. W. (1987). Hierarchical credit allocation in a classifier system. In 10th *International Joint Conferences on Artificial Intelligence (IJCAI'87)*, pages 217–220, Milan, Italy.

[Wilson, 1994] Wilson, W. (1994). ZCS: A zeroth level classifier system. *Evolutionary Computation*, 2(1):1–18.

[Wilson, 1995] Wilson, W. (1995). Classifier Fitness Based on Accuracy. *Evolutionary Computation*, 3(2):149–175.

[Wilson, 1996] Wilson, W. (1996). Explore/exploit strategies in autonomy. In *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior (SAB96)*, pages 325–332.

[Wilson, 2000] Wilson, W. (2000). Get real! XCS with continuous-valued inputs. *Lecture Notes in Computer Science*, 1813:209–219.

[Yu and Terzopoulos, 2007] Yu, Q. and Terzopoulos, D. (2007). A decision network framework for the behavioral animation of virtual humans. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 119–128. Eurographics Association.