AISB Journal

The Interdisciplinary Journal of
Artificial Intelligence and the Simulation of Behaviour

Volume 1 – Number 3 – July 2003

The Journal of the
Society for the Study of Artificial Intelligence
and the Simulation of Behaviour
http://www.aisb.org.uk

Published by
The Society for the Study of Artificial Intelligence and Simulation of Behaviour

http://www.aisb.org.uk/

ISSN 1476-3036

Contents

AISB Journal **1**(3), © SSAISB, 2003
http://www.aisb.org.uk

Intelligence as Physical Computation

Sunny Bains

Intelligent and Interactive Systems, EEE Department, Imperial College London
Exhibition Road, London SW7 2BT, UK
Department of Design and Innovation, The Open University
Milton Keynes MK7 6AA, UK
sunny@sunnybains.com

Abstract

In this paper, we describe a model of physical computation that allows us to understand how embodied intelligent agents can simultaneously be considered to be objects operating under the laws of nature (or physics) and information processing devices. Using this type of analysis has two major advantages. First, it allows us to make concrete those issues relating to the relative merits of using analogue values or symbolic representations such as numbers: our analysis allows this longstanding point of contention in artificial intelligence to be transformed from a question of philosophy to one of physics. Second, it gives us a framework that should eventually allow us to produce design rules that will enable our artificially-intelligent (physical) agents to make better use of influences from the physical environment. As should be expected, for well-defined tasks, the results of this analysis are identical to Shannon's information theory. However, for large, multi-functional systems with ill-defined roles, the new model provides a novel way of thinking about the complementarity of hardware and software.

1 Introduction

In this paper, we take a fresh look at what intelligence is, where it fundamentally comes from, and how this affects the way we think about building artificially-intelligent agents. Note that this model is exclusively directed at the goal of understanding embodied intelligence and, particularly, intelligence that is based in the natural or real (as opposed to mathematical) world. In this sense, though it applies to any real machine that can perform mathematics, the model is *least* useful for those whose goal is mathematical manipulation rather than the processing of signals from the outside world in order to produce intelligent actions.

This research has come about through a desire to bridge a gap between two existing fields. First is theoretical work that proves that analogue computation, in the form of recurrent analogue neural networks (RANN), can be super-Turing in nature (Siegelmann, 1999). Siegelmann demonstrated that, if allowed to take continuous rather than discrete weights, recurrent neural networks could perform functions that are theoretically impossible using Turing machines. The problem with this model is that it is entirely mathematical: in it, there is no interface to the real (physical) world, so assumptions she made (about continuity, for instance, or noise) have no clear way of being validated. Likewise,

studies that look at how noise can degrade the abilities of such machines (Maass, 1997) do so from a theoretical rather than physical perspective.

The link between the theoretical and the physical is important because the brain looks very much like a RANN. If Siegelmann's conclusions were applied to the brain, then Roger Penrose's (1989) assertion that human-like intelligence could not be performed on Turing machines could be correct (though for the wrong reasons).

It is important, at this point, to distance the arguments made in this paper from the debate Penrose famously started concerning physics, artificial intelligence and computationalism. Using Gödelian arguments (Gödel, 1931), he pointed out that certain propositions are undecidable in Turing machines (as they are in all such mathematical systems) based on the axioms inherent in those systems. He also claimed that microtubules in the brain (Penrose, 1997) had quantum-mechanical properties that were both non-Turing-computable and potentially important to intelligence. These claims have generally been disputed from both computational and physical points of view. First, it has been argued that the specific Gödelian limitations are not, in fact, in conflict with human intellectual abilities (e.g. Laforte *et al.*, 1998). Second, the timescales related to quantum decoherence in microtubules have been shown to be so different from those relevant to the brain that classical (rather than quantum) neural behaviour has been proposed as the more appropriate model (Tegmark, 2000).

Instead, this paper is more concerned with issues such as those succinctly outlined by Dreyfus in the 1970s (Dreyfus, 1972). In his book *What computers still can't do: A critique of artificial reason*, Dreyfus explains why the formalization of a physical process is not the same as the process itself (Chapter 5). With this paper, we take a more engineering-based approach to his philosophical questions. We ask, if the behaviour of physical systems cannot be replicated using Turing machines, how can they be replicated?

Back to the technological gap that we are trying to bridge. On one side we have the work done in theoretical computer science (by researchers like Siegelmann), while on the other we have the field of neuromorphic engineering: where engineers try to structure their machines, often including the hardware, in a brain-like way. Carver Mead's analysis of the power-efficiency of analogue computation (e.g. Mead, 1989), particularly for neural networks, has been extremely important in this regard. He demonstrated that by exploiting, rather than fighting against, the intrinsic physics of electronics, analogue circuits could be orders of magnitude more efficient than their digital counterparts. Leon Chua's cellular neural network (e.g. Chua, 1998), a device that is digitally programmable but can perform complicated nonlinear operations during the analogue transient—the "switching" time to go from one stable state to the next—is an apt demonstration of Mead's point. Not only is the device far lower-power than the equivalent image-processor, but it is also up to orders of magnitudes faster (depending on the algorithms implemented). Nabil Farhat's optical implementations of biological neural models (e.g. Farhat, 1997 and Farhat and Wen, 1995) not only show the utility of analogue networks, but their potential complexity. From conceptually very simple optical and/or analogue circuits, he has obtained behavior that—within a single system—varies between periodic, m-periodic (repetition of a pattern of m beats), psuedo-periodic (qualitatively periodic, but not strictly so quantitatively), and chaotic (with a fractal set as an attractor) output.

Though these researchers never made any explicit claim that their systems were computationally superior, Siegelmann's model suggested they might be. Thus, it seemed logical to find an approach that would bridge the gap between the two. To date, Siegelmann and others (e.g. Blum *et al.*, 1989) have worked with notions of super-Turing or hypercomputation: forms of computation that can perform functions theoretically impossible with conventional Turing machines, such as functions based on non-computable algebras

or that are non-recursive. In particular, they have concentrated on computation over the reals. This theoretical approach, though fruitful, does not tell us what is possible with real machines. Our new model, on the other hand, was designed with the following intentions: a) that it should have a clear interface with physics; b) that it have a clear application to artificial intelligence in the simulation-of-behaviour sense (as opposed to the solution-of-arbitrary-mathematical-problems sense); and, c) that it allow for an understanding of the differences and relationships among various different types of machine.

Note that, though the model presented here does all of the above—including providing an interface with physics (in that it can be interrogated using what we know of the physical laws)—physical interrogation is beyond the scope of this paper (but is at an advanced stage).

The paper is structured as follows. In Section 2 we discuss the Turing machine and why, whatever its comparative computational power, it is an inadequate model for the embodied intelligence task. In Section 3 we introduce the basis of the new model, showing some important variations on how it can work in Section 4. The basics of how the model can be used as an interface between the computational and physical paradigms are explained in Section 5, with a discussion of the model—with particular reference to neuromorphic engineering—in Section 6. Finally, in section 7, we will describe current avenues of this research that are beyond the scope of this paper, and future avenues that may be productive.

2 The Turing machine as a model of intelligence

When Turing wrote his original paper describing the thought experiment that became known as the Turing machine, he described it as a kind of automatic version of doing what people do when they manipulate numbers and symbols on paper in an intelligent way (Turing, 1936). The machine had an arbitrarily long tape from which it could read as many symbols (from a finite set) as needed, means to read and manipulate those symbols (the read/write head), a finite set of states where it could store information about what had gone before, and a set of rules that governed what it should do in the event of various combinations of input and state.

Critically, the input and output are explicitly constrained to be symbolic. Thus, the machine is unable to interact directly with the environment: it must always receive a set of symbolic inputs from sensors, which have to perform some type of transformation into the relevant symbol space (such as digitisation), and must produce its output as a set of control symbols that can then be used to affect the real world.

This fact is critical, because it means that there is no way that a Turing machine can perform any kind of natural behavioural intelligence—where it responds to some external force or signal by moving or changing state—it can only perform the intelligent manipulation of symbols. It is, therefore, not just wrong but essentially *meaningless* to speculate on the ability of Turing machines to be able to perform human-like intelligence tasks. In any real machine, but particularly those designed to interact with the environment, the outer shell (body, sensors, actuators) must be, by definition, entirely analogue. This is because the signals that they deal with (from the outside world) are analogous to the real physical values in question (or, more precisely, they *are* the real values). For the machine to work, at some stage after this outer-shell has been breached, an analogue-to-digital (a/d) conversion step must take place, thus allowing the digital computer to do whatever processing is required. The same, in reverse, is true for actuation.

Given this, every robot (and, to a lesser extent, every computer) is a hybrid machine:

part analogue, part digital. In the following sections, a crucial question in the design process of any machine, but particularly a machine intended to be tightly-coupled to its environment, is highlighted: the location of the boundary between these two parts. In particular, we believe the virtual interaction variation of the model gives some insight into this question though, as yet, it falls short of providing specific design rules.

Note that the above argument is not anti-computationalist in the traditional sense. Rather, the intention here is to point out that computationalism as it is normally discussed is moot. Since all robots must be hybrid machines (and, therefore, not *necessarily* constrained by the well-known limits of Turing computation), setting up the Turing machine as the only route to artificial intelligence is, in effect, setting up a straw man. From a theoretical perspective the question is whether the hybrid machine has the computational power to do the job. From an engineering perspective, the question is how such a machine can be designed to do the job *most efficiently*.

3 The physical computational model

Here we present a mathematical model of a potentially-intelligent, embodied, adaptive, physical system: one that includes mechanisms that can be interpreted as allowing learning via experience of the environment and action based on that experience. In Section 5 we consider the elements that are affected by physical constraints (i.e. the constraints of the real physical world as opposed to specific engineering constraints), but for now we simply lay out the mathematical model.

A *system* is here defined as an identifiable collection of connected elements. A system is said to be *embodied* if it occupies a definable volume and has a collective contiguous boundary. The matter, space and energy outside the boundaries of the embodied system are collectively called the *environment*.

A *sensor* is any part of the system that can be changed by physical influences from the environment. These, which include any or all forces, fields, energy, matter, etc. that may be impinging on the system, are collectively called the *sensor input* ($x \in \mathcal{X}$), even where no explicitly-defined sensors exist. Though represented by a single variable, the sensor input may in fact consist of many different *sensor modalities* (each influenced by a different type of force or energy).

Resulting external physical changes to the embodied system, (emission of light, movement of a limb, etc.), are collectively called the *actuator output* ($y \in \mathcal{Y}$) of this function. An *actuator* is any part of the system that can change the environment. A coupled pair of sensor input and actuator output may be described as a *behaviour*.

Let us define $\mathbf{G}$ as the *intelligence function* performed by the embodied system, mapping the input to the output. Where t is time, and $\delta t > 0$ but arbitrarily small (in other words, time may be continuous—whether it is or not is a physical question that will be addressed in a future paper), we have:

$$\mathbf{G}_t(x_t) \rightarrow y_{t+\delta t}$$

It is important to note here that, for our purposes, $\mathbf{G}_t$ can only be considered to cause an immediate actuator output (change that may effect the environment) as a result of an immediate actuator input (physical influence from the environment). It cannot be considered to implement any kind of plan over time, like commanding robot arm to move through a particular trajectory. Instead, the plan is carried out through $\mathbf{G}$ itself changing with time. $\mathbf{G}$ is altered by any internal changes to the system caused by the input (flow of a current inside a wire, charging of a capacitor, shifting of weight, etc.). The *learning*

function, $\mathbf{L_G}$, determines how $\mathbf{G}$ changes with time: $\mathbf{L_G}(\mathbf{G}_t, x_{t-\delta t}) \rightarrow \mathbf{G}_{t+\delta t}$. This process is called *adaptation*.

This is a subtle, but important, difference in the way we think about how machines work. Were the intelligence able to issue a command to be followed by an actuator over time, some controller would have to be at work in the actuator to make sure that the command was carried out. This is fine, but in our system this controller is considered to be *part* of the intelligence function.

Now, let x' be the output from the environment to the system, and y' be the system's input to the environment: the impact of the system's behaviour on its surroundings. Where $\mathbf{E}$ is the *environment reaction function* mapping the input to the output:

$$\mathbf{E}_t(y'_t) \rightarrow x'_{t+\delta t}$$

There is also an *environment learning function*, $\mathbf{L_E}$, equivalent to $\mathbf{L_G}$, that determines how $\mathbf{E}$ changes with time.

4 Categories of physical interaction

4.1 Real interaction

The interaction between $\mathbf{E}$ and $\mathbf{G}$ may be considered to fall into one of two classes. *Real interaction* is a pure physical process in which embodied intelligence *co-evolves* with its environment: such that the two functions are dependent only on initial conditions, their governing functions ($\mathbf{L_E}$ and $\mathbf{L_G}$), their interactions with each other, and time. Thus, any adaptation of the embodied system is in direct response to its environment and nothing else (and vice versa). See Figure 1. This type of interaction requires that $x = x'$ and $y = y'$. Consequently, the domains/ranges of the two systems must be the same.

In the real-interaction scenario, as well as being a function, $\mathbf{G}$ may also be considered a description of the system's *instantaneous physical state* at the arrival of the input stimulus. It is important to note that the state is here defined as not only specific parameters that can be measured instantaneously (speed, position, etc.), but also all associated rates of change. For example, two balls—one at rest and the other falling under gravity—would not be considered to have the same intelligence function even if they were identical in all other ways. Instantaneously they might look the same, but with associated rates of change taken into account, they are clearly not.

This should also highlight how intelligence in the physical sense is different from our idea of intelligence in computers normally. With conventional computers, we consider intelligence functions on *subsets* of stimuli and represented as abstractions. This makes them implementation independent: an adder can be built in many different ways with many different materials, but intelligence is required to determine what this intelligence function actually is and to *interpret*—which we can define as weeding out the relevent from the irrelevent—the results. With physical computation, the function and implementation are one and the same.

In the real-interaction case, implemented in the physical world, various constraints may be imposed. These include the following:

- The combined values of certain physical parameters of the two systems may be conserved (e.g. conservation of energy).

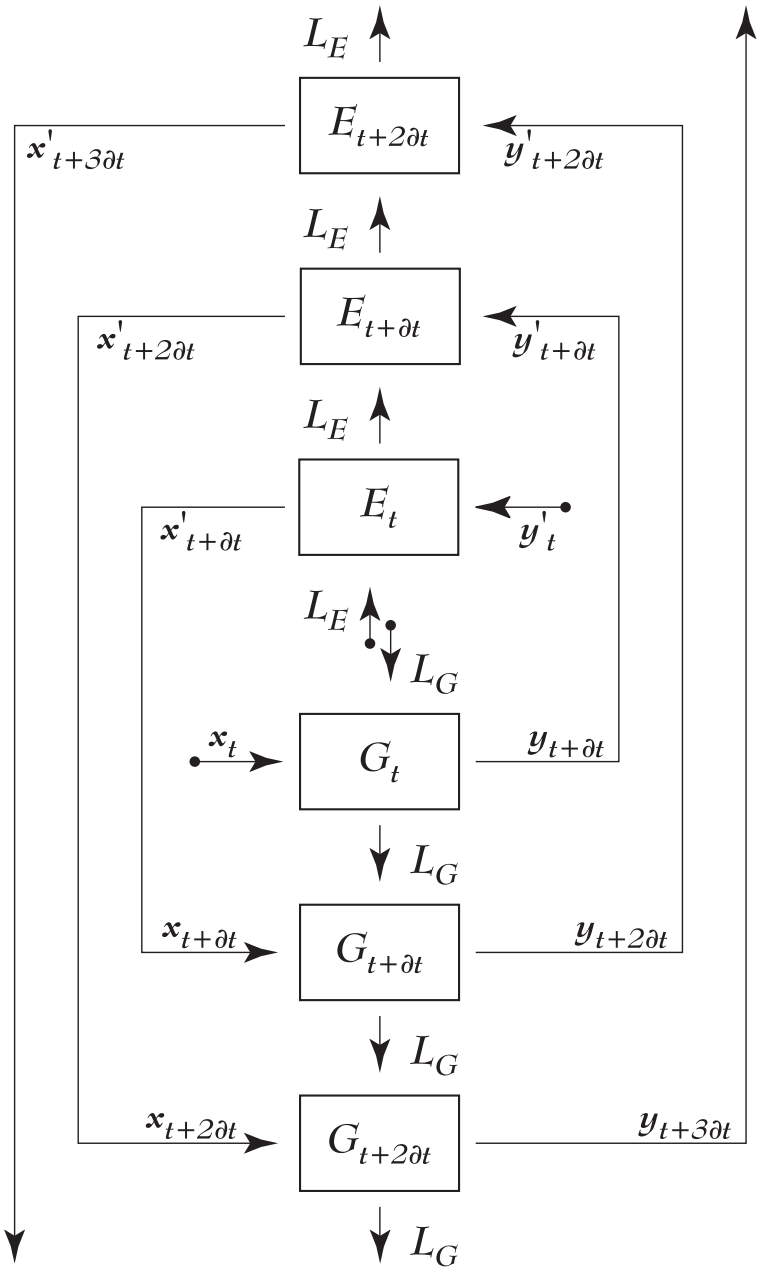

Figure 1: Here an object is evolving under physics. **G** is the function that the object performs on the inputs (x) it gets from the outside world, which determine how it will impact the outside world (y). Likewise for **E**, which is the function performed by the outside world on the input from the object (y'), producing the output (x'). These two functions are carried out in parallel. Though labelled differently, x and x' must be the same for real interaction to take place, and likewise for y and y': thus the domain of each function must be the same as the range of the other.

- Changing functions and variables in the model may be constrained to evolve in certain ways: i.e. they may be constrained to vary continuously or to have particular allowed values.

- Since the only driving mechanism available is the function collectively known as *the laws of physics* (as they exist rather than as we understand them), $\mathbf{L_P}$, the constraint that $\mathbf{L_E} = \mathbf{L_G} = \mathbf{L_P}$ may be imposed.

  In Section 5, we will consider some potential specific constraints of the laws of physics.

4.2 Virtual interaction

The second class of interaction to be considered is *virtual interaction*, which may be mediated by *symbolic representation* and *communication* (thus allowing the domains/ranges to be mismatched). Here, we define a *symbol* using Turing's 1936 definition: a letter or sign taken from a finite alphabet to allow distinguishability. We define communication using Shannon's communication theory: the sending of a *message* to a *receiver* via a (potentially) *noisy channel* (Shannon and Weaver, 1949). See Figure 2.

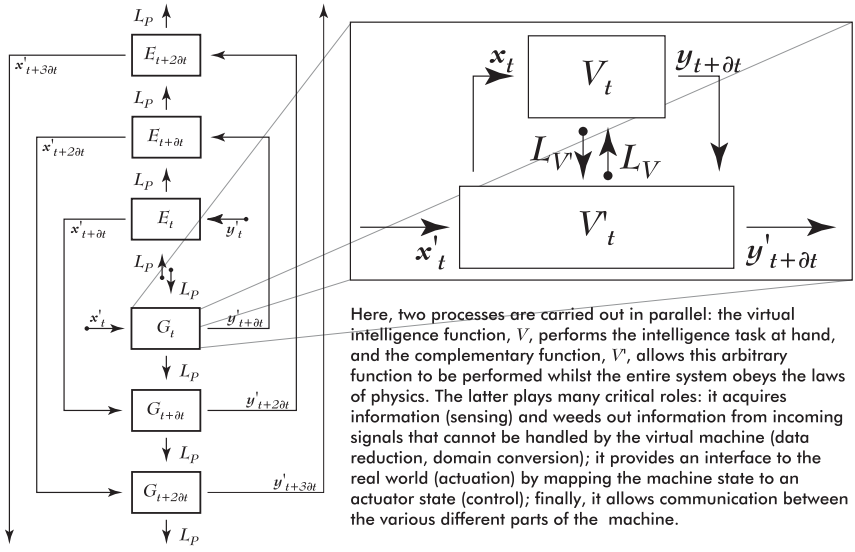

Figure 2: In deterministic virtual interaction, real interaction is also taking place. However, the output from the environment is not the same as what is being input to the function that we deem intelligent: the virtual machine with function $\mathbf{V}$. For this to be the case, we model the interaction as including a complementary function ($\mathbf{V}'$), that filters the input from the outside world, feeds it into the virtual machine, processes the remainder, and combines the virtual machine result with its own to produce the final output. The virtual machine and its function are normally analyzed by computer scientists, whereas the complementary functions are the province of electrical, electronic, mechanical, and other engineers.

If we define $\mathbf{V}$ as the *virtual intelligence function*, analagous to $\mathbf{G}$ but for the virtual case, some major constraints are lifted, with important consequences. First, x is no longer

constrained to be equal to x' (likewise for y and y'). What this means is that the inputs and outputs may not be considered in their totality, but selectively (entire modalities may simply be excluded by x and y, or specific ranges within specific modalities may be excluded). This can be considered in two ways. From a design perspective, it means that behaviours (sensor-actuator pairs) may be considered identical even if the global inputs and outputs (the inputs and outputs taking all possible modalities into account) are very different.

For example, when the letter A is typed into an laptop computer, how hard the key is pressed (within a range) is irrelevant. Soft A and hard A are considered, within the virtual environment, to be identical inputs. Likewise, the brightness of the screen is irrelevant to the meaning of the letter A when it appears before the user: whether or not our laptop is in power-saving mode does not affect our perception of the output here. So we have virtual sensor-actuator pairs (letter is typed, appears on screen) that can be very different physically but are considered to be the same behaviour virtually. In this scenario, there are fewer sensor modalities available to the embodied intelligence than there are actual modalities of physical influence coming from the environment.

As a result of this, the actual output from the environment may be very different from the input received by the virtual embodied intelligence. They may be very different because they are not allowed to affect the working of the virtual machine at all (just as how hard a key is pressed is information that is not available to the computing machine within the laptop). Or they may be represented very differently. For instance, temperature, which may be continuously varying in the environment, may be represented as a number with just one decimal place in the machine. Mathematically, in either of these cases, the domain and range of the two functions may be different.

In order for all of the above to be true, a new function must be defined: the *complementary function* $\mathbf{V}'(x'_t - x_t)$ that ensures that, together with the intelligence function, the entire system obeys the laws of physics. The existence of this function can be considered to be a test of whether a system is virtual or not.

Another important constraint that is lifted is that $\mathbf{E}$ and $\mathbf{V}$ need not have any kind of conserved relationship, and $\mathbf{L}_{\mathbf{V}}$ need not be the same as $\mathbf{L}_{\mathbf{E}}$. Because only range/modality subsets of x' and y' attach to $\mathbf{V}$ and $\mathbf{L}_{\mathbf{V}}$, the intelligence and adaptation functions are partially de-coupled from the environment. With the right machine and choice of sensor modalities, arbitrary choice of $\mathbf{V}$ and $\mathbf{L}_{\mathbf{V}}$ may be made.

It is important to note that this arbitrary choice *depends* on the selection of inputs/outputs, since a real interaction (with adaptation function $\mathbf{L}_{\mathbf{P}}$) *must* be taking place at the same time as this virtual interaction. Thus, it is only because of $\mathbf{V}'$ and $\mathbf{L}_{\mathbf{V}'}$ that such freedom is allowed for $\mathbf{V}$ and $\mathbf{L}_{\mathbf{V}}$.

4.3 Non-deterministic interaction

The argument so far assumes (as is generally assumed in all branches of physics, except quantum mechanics, which will be discussed in more detail later) that the physical evolution taking place is both a causal and a deterministic process. In this context, causal means that the state of the two systems $\mathbf{E}$ and $\mathbf{G}$ at a given time t is the direct and only cause of its state at time $t + \delta t$. In other words:

$$(\mathbf{E}_t, \mathbf{G}_t) \rightarrow (\mathbf{E}_{t+\delta t}, \mathbf{G}_{t+\delta t})$$

Deterministic means that the state at time $t + \delta t$ can be predicted from that at t. Non-deterministic means that it cannot. Note that there is an ambiguity inherent in this

definition. It is not stated by whom or by what this prediction can be made, nor in what circumstances. It may be unpredictable in principle because the process has some kind of random element, one not following any principle or order, embedded in it. Or it may be non-deterministic in practice: because insufficient information is available to make a reliable prediction.

It may seem that an interaction may not be both causal and non-deterministic. However, from the perspective of conventional quantum mechanics, the two are compatible in the following sense: physics (in the form of the wave equation) causes a particular set of outcomes to be possible, but which of these outcomes actually takes place is not determinable in advance. This is the definition of a stochastic process. The word random, left undefined in the last paragraph, can be more clearly understood in this context. If the process of "choosing" between different possible outcomes is not determined by physics, then it must be determined by something outside or above physics: meta-physics.

Two examples of *in-practice* non-determinism may make clearer the distinctions between this and the *in-principle* variety described above. First, a chaotic system might be *practically* non-deterministic to an observer if it were impossible to get infinitely precise information about it: however, there may be no principle that says that such information could not be made available. Second, a quantum-mechanical system might be considered unpredictable by scientists because the required properties and variables to make the prediction cannot be measured without changing them.

As the physical model presented in this thesis is not based on the manipulation of information or prediction based on a model, the *in-practice* scenario does not qualify as non-deterministic. Consequently, the model described in this section only applies if the *in-principle* variety of non-determinism is true. This distinction becomes important when the issue of how the mathematical model relates to real physics is considered.

4.3.1 Model of stochastic interactions

Mathematically, a causal, deterministic process can be represented by a one-to-one mapping from the input to the output: for a given set of conditions, only one outcome is possible. Both the real and virtual interactions described in the previous sections are based on this type of mapping. A stochastic or non-deterministic interaction, on the other hand, must be represented as a one-to-may mapping. In this case the output (x' or y) cannot be represented by a single value. Instead it must be taken from a set. In principle, this set may have any cardinality (with the possible exception of being empty): it may be finite, denumerable or non-denumerable. In Figure 3, for simplicity, the set of possibilities is kept to just two.

As can be seen in the figure, there is not (by definition) a pre-determined timeline. Specifically, knowing the state of one system at one time no longer uniquely implies the state of the other at that time, nor its own state any time later. This can be understood by following the various allowed evolutionary pathways for the two systems. As time elapses, the number of states that each system may have evolved into increases. This may be considered a type of divergence, in that it directly prevents tight coupling between the two systems.

To allow the mathematical model to take this type of evolution into account, a new *random variable*, z, can be defined that chooses which of the possible physical outcomes occurs: this variable is entirely independent of x, y, x', and y'.

Such a scenario may be considered to be a type of *double* virtual interaction. Functions $\mathbf{G}$ and $\mathbf{E}$ are constrained (by definition) to only act on input from each other (x, y, x', and y'). And yet, somehow, there is a function in existence that operates on z. As our model

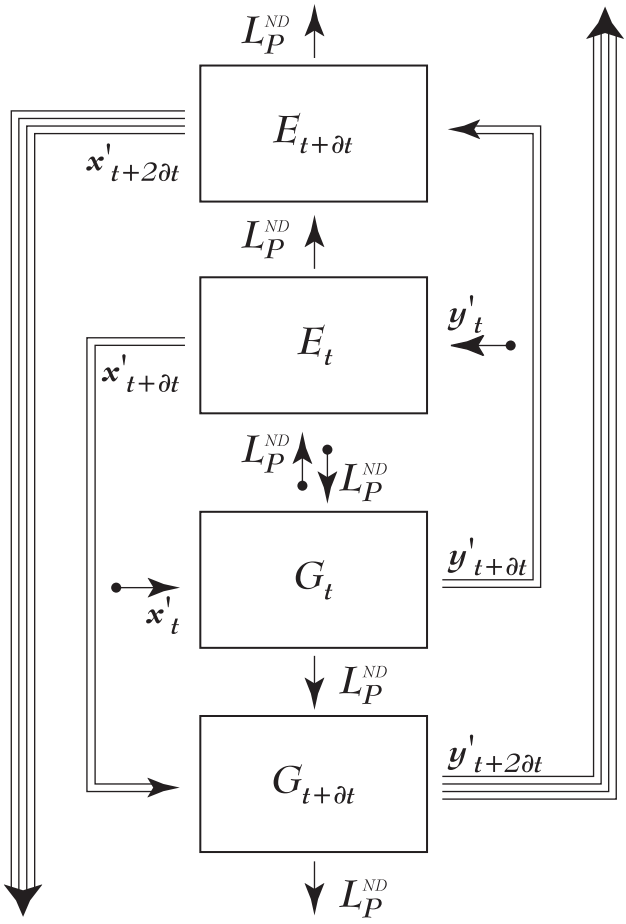

Figure 3: In a non-deterministic universe, more than one outcome may be possible for a given physical event (the non-deterministic laws of physics are used here: L_P^{ND}). Two are shown here for simplicity. Thus, the coupling relationship between the two systems may change over time depending on which course (path) events take. Since the choosing of the path is metaphysical (not produced by object or environment), such an interaction cannot be considered real as it has been defined here.

is defined, the only way this is possible is if $\mathbf{E}$ and $\mathbf{G}$ are mapped to virtual functions $\mathbf{V^E}$ and $\mathbf{V^G}$ and their complementary functions, $\mathbf{V'^E}$ and $\mathbf{V'^G}$, are considered to keep track of z (see Figure 4). Here, $\mathbf{V^G}(x) \rightarrow \mathcal{Y}$, and $\mathbf{V'^G}(\mathcal{Y}, z) \rightarrow y'$. Crucially, z is metaphysical here: it's value is determined by some process that is not entirely dependent on, or related to, any of the physical laws or variables.

Given this analysis, only deterministic physical processes can be considered to fall into the class of *real interactions* as defined above.

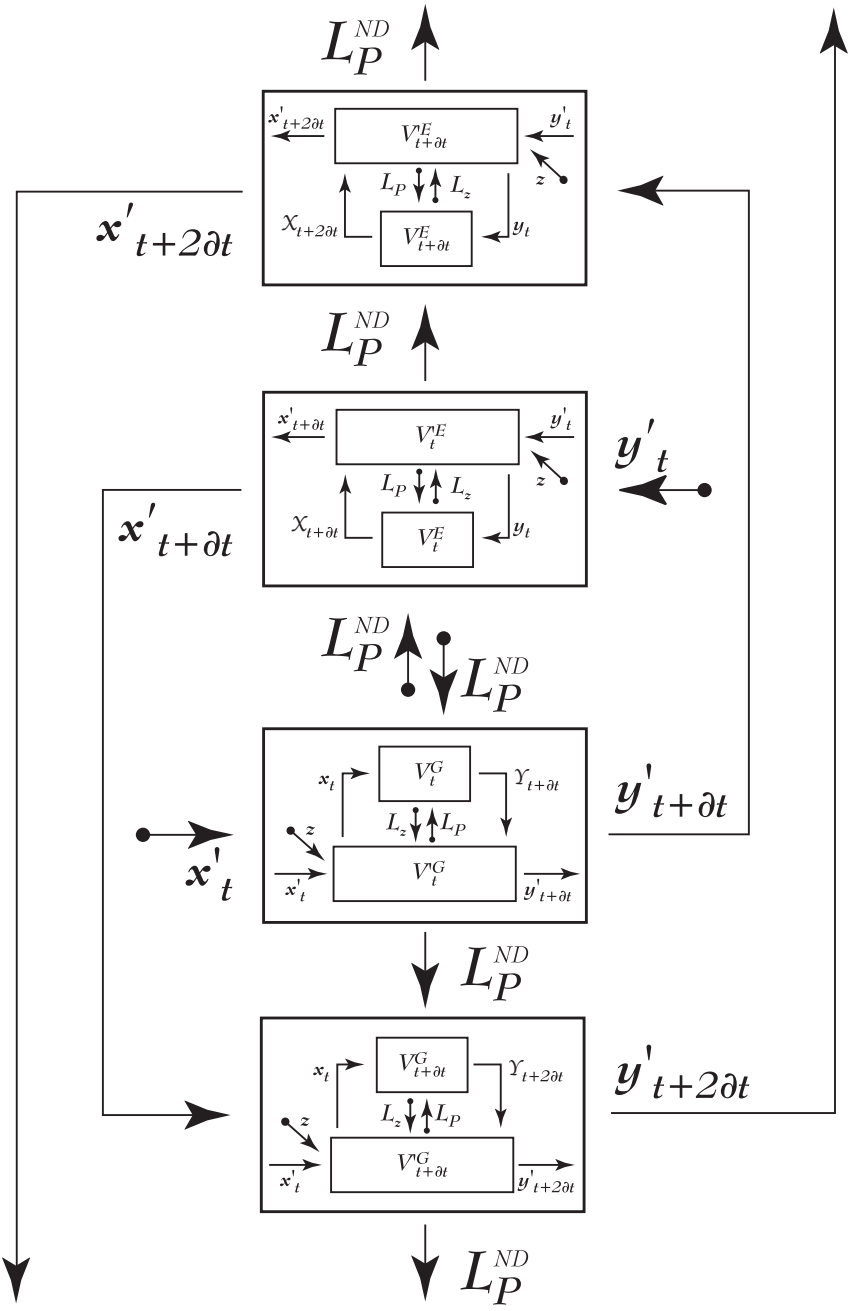

Figure 4: Instead, the non-deterministic case can be considered to be a kind of double virtual interaction, where the virtual machine is that obeying the deterministic physical laws that produce a set of possible outcomes, and the complementary function chooses from this set based on some metaphysical random variable, z.

4.4 Real versus virtual interaction

To summarize this section, it is worth going back to our laptop example and comparing the real and virtual interaction analyses. In the former case, the machine may be considered just like any object. It is something that moves under gravity, has resistance, gives off heat, makes noise when touched etc.. It can be used as a paper-weight or a doorstop. All of these uses of the machine are covered. In the virtual analysis, only the tapping of the keys and the pixels on the screen are considered to be part of the machine's intelligence function: $\mathbf{V}$. All other aspects of its behaviour, which combine to give it the global function $\mathbf{G}$, are included in $\mathbf{V}'$.

Thus, for simulated robots, artificial life, and other applications where the interaction is purely symbolic and has no real meaning (apart from for display purposes) outside the machine, $\mathbf{V}$ is all that's important: the complementary function is irrelevant (as it is normally treated in computer science). For real robots in a real world, however, $\mathbf{G}$ is critical. Thus, $\mathbf{V}'$ must be taken into account when designing and building them.

5 The interface with physics

As described so far, the model is entirely general, in that we have not assigned the various functions or variables as belonging to any specific sets. It is only with this assignment that the representational and computational power of the model can be determined and (as required) compared to other computational models (such as the Turing machine). In order to provide the bridge between the theoretical and the physical, these assignments must come from our understanding of the physical laws. The relevant correspondences are outlined below.

The first important physical question that must be answered in order to allow comparison between various models has already been alluded to in Section 4: this is the issue of whether physics is entirely deterministic, or whether it is not. As discussed earlier, this information allows us to know whether real interaction actually exists, and therefore which model should be used.

Second, x, x', y, y', are place holders for the multidimensional space that includes all the physical state variables, and t is the placeholder for time. From a mathematical perspective we can ask two very simple questions about these variables that allow us to begin to understand their representational power. Are their sets finite or infinite? If infinite, do they have cardinality $\aleph_0$ or $\aleph_1$? These questions are crucial, because the answer determines whether or not the physical computation is less powerful, more powerful, or as powerful as (for instance) the Turing machine.

For instance, if it turns out that $\mathbf{G}$ can map a continuous variable onto a continuous variable ($|\mathcal{X}| = |\mathcal{Y}| = \aleph_1$) then this is the equivalent of a machine with an infinite symbol set and an infinite rule book: this is, specifically, not allowed with a Turing machine and represents the super-Turing case (it potentially contains all the mappings available to the Turing machine as well as others). For $|\mathcal{X}| = |\mathcal{Y}| = \aleph_0$, and $|\mathcal{X}| = |\mathcal{Y}| = N_{MAX}$ (a finite set), the machine would be Turing-equivalent, or sub-Turing, respectively.

With some rearrangement, the questions about cardinality can be turned into physical ones and then fed back into the model. First, are the multi-dimensional physical state variables and time—or, more simply, is space-time—continuous, discrete (i.e. there is a minimum δx or δt), or arbitrarily discrete (i.e. there is no specific minimum δx or δt but the variable is still not defined as continuous). Second, are the physical systems in question bounded or unbounded? For this latter question, in the case of the embod-

ied intelligence, the answer is given in the definition: it *is* bounded. In the case of the environment, however, the question is still not decided among physicists.

We will leave this discussion here, as a full explanation of the relationships between the various physical and mathematical options is beyond the scope of this paper. However, we hope that it is, at least, clear that an interface between the theoretical and physical paradigms does exist here.

6 Discussion

In the brief discussion of comparisons above, a practical matter is not made explicit. The question we are asking is whether or not $\mathbf{G}$ may be replaced with a Turing machine. In fact, as discussed in Section 2, we know it cannot be, because there is no mechanism for getting information in and out of the machine from the real world. Thus, it must be the complementary function, $\mathbf{V}'$, that allows us to feed our Turing machines (or digital computers) with the symbols they need, and to use the resulting symbols to produce an output. Thus, we can explicitly say that any symbolic interaction is, by definition, a virtual interaction, and the complementary function is crucial to its success. What, in practice, is $\mathbf{V}'$? It is the machine's analogue sensors and actuators and enablers: mechanics, optics, hydraulics, heat-sinks, etc..

The same thing might also be said of our bodies, which not only provides our conscious mind with sensors and actuators, but also the un/subconscious mind which is not normally considered in AI. Other bodily systems—the limbic system, the spinal chord, even basic organs like the heart and lungs—all contribute to our behaviour to a greater or lesser extent. Using the virtual interaction model, we can see that what many people consider to be artificial intelligence, particularly purely software-based approaches, concerns only $\mathbf{V}$ and not $\mathbf{V}'$.

What is interesting here is not only that this $\mathbf{V}'$ exists, but that it may be as important as $\mathbf{V}$ for some applications.

This fact is increasingly being recognised by roboticists. For instance Williamson, who worked on Rodney Brooks "Cog" project, was charged with engineering the robot's limbs in such a way as to ease both the information-processing and energy burden they represented for the machine as a whole. A mechanical engineer by training, he designed Cog's arms and wrists so that they were compliant (Williamson, 1995) and could respond *mechanically* to changes in the environment rather than purely through conventional sensor-processor-actuator loops. The result was not only a more mechanically-efficient and natural-looking movement, but considerably lower computational overheads. Others interested in exploiting a balance between information processing and physical (in this case, mechanical) computation include Lungarella and Berthouze (2002), who have looked at how temporarily restricting the degrees of freedom of a mechanical system can improve a robot's ability to learn how to manipulate it, and Pfeiffer who gives numerous examples of the importance of mechanical design to machine intelligence (Pfeifer, 2002).

This balance may also be said to be the concern of the neuromorphic engineering field pioneered by Mead. This work often goes much further, blurring the interaction boundaries. Projects like Harrison and Koch's all-analogue fly vision system and robot controller (Harrison and Koch, 2000), Hasslacher and Tilden's analogue walking robots based on the nervous systems of small animals (Hasslacher and Tilden, 1995), or Lewis's bipedal robots based on central pattern generators (Lewis *et al.*, 2001), all have in common that there cannot be said to be a clear line between sensor, processor, and actuator. Indeed, there cannot be said to be a clear line between software and hardware. Whether their

systems can also be considered to perform real interaction or not is complex (and will not be considered here), but their machines certainly seem to address the complementary function more fully.

Finally, one could also argue that full understanding of real interaction, virtual interaction, and the complementary function might go some way to addressing Brooks' recent question about the relationship between matter and life (Brooks, 2001). In essence, we may just need to go further to try to understand a whole brain and body (both $\mathbf{V}$ and $\mathbf{V}'$) in order to understand how a creature works, rather than considering mind alone.

7 Current and future work

From the description above, several projects suggest themselves: some of which are currently underway and others that might usefully be done in future. The first of these, currently in progress, is the analysis of the model under constraints as suggested by current (and often contradictory) interpretations of the physical laws, and comparisons with the Turing model given the options that arise from these. In particular, issues related to quantum mechanics are problematic. Whether the universe may be considered discrete or continuous is not a trivial problem: there is no consensus here. Likewise, there are still many who believe that the current consensus in quantum-mechanics—of which it may be said that the theory is sound, but the philosophical underpinnings not—will eventually have to change. We are currently developing a map of these ideas so that, as the physics develops, its impact on the question of intelligence can be seen clearly. This work is currently being written up.

We are also trying to identify those applications of intelligence where a mismatch in representational power may be important. Clearly, information-theoretic approaches are appropriate in many engineering scenarios: today's engineering is based on such approaches. We are currently drawing up a simple specification, in terms of the types of functions that may be important and the type of stimuli that may need to be represented, that should allow engineers to understand where a more physical approach may be warranted. This work is also at the draft stage.

As a longer term goal, we would like to be able to help the engineer who, using the test outlined about, has determined that the conventional design approach is inappropriate for the task in hand. This involves the development of a set of design rules that would determine how analogue to digital conversion layers are placed in a given system: i.e. how to correctly balance the analogue and digital processes in order to maximise efficiency for a given task. Note: in some cases it might be expected that *no* conversion is the best option, thus suggesting an analogue-only system.

Acknowledgements

The author would like to acknowledge: her supervisor at the Open University, Jeffrey Johnson; the help of the reviewers; Christof Koch of the California Institute of Technology for important discussions and reading the manuscript; her colleagues at Imperial College; and numerous other scientists and engineers who have contributed to this work through inspiration, discussion, and debate.

References

Blum, L., Shub, M., and Smale, S. (1989). On a theory of computation over the real numbers; NP completeness, recursive functions and universal machines. *Bulletin of the American Mathematical Society*, **21**(1), 1–46.

Brooks, R. (2001). The relationship between matter and life. *Nature*, **409**, 409–411.

Chua, L. (1998). *CNN: A Paradigm for Complexity*. World Scientific.

Dreyfus, H. (1972). *What computers still can't do: A critique of artificial reason*. MIT Press.

Farhat, N. (1997). Cognitive networks for ATR: The roles of synchronicity, bifurcation and chaos. Technical report, Office of Naval Research. Final Report.

Farhat, N. and Wen, Z. (1995). Large-scale photonic neural networks with biology-like processing elements: the role of electron trapping materials. *Proceedings of the SPIE*, **2565**.

Gödel, K. (1931). *On formally undecidable propositions of Principia Mathematica and related system*. Dover.

Harrison, R. and Koch, C. (2000). A silicon implementation of the fly's optomotor control system. *Neural Computation*, **12**(10), 2291–2304.

Hasslacher, B. and Tilden, M. (1995). Living machines. *Robotics and Autonomous Systems*, **15**, 143–169.

Laforte, G., Hayes, P., and Ford, K. (1998). Why Gödel's theorem cannot refute computationalism. *Artificial Intelligence*, **104**(1-2), 265–286.

Lewis, A., Hartmann, M., and Etienne-Cummings, R.and Cohen, A. (2001). Control of a robot leg with an adaptive aVLSI CPG chip. *Neurocomputing*, **38-40**(1-4), 1409–1421.

Lungarella, M. and Berthouze, L. (2002). Adaptivity through physical immaturity. Oral presentation at 2nd International Workshop on Epigenetic Robotics.

Maass, W. (1997). Analog neural nets with gaussian or other common noise distributions cannot recognize arbitrary regular languages. In *Electronic Colloquium on Computational Complexity Report*. TR97-052.

Mead, C. (1989). *Analog VLSI and Neural Systems*. Addison Wesley.

Penrose, R. (1989). *The Emperor's New Mind*. Oxford University Press.

Penrose, R. (1997). *The Large, the Small, and the Human Mind*. Cambridge University Press.

Pfeifer, R. (2002). On the role of embodiment in the emergence of cognition: Grey Walter's turtles and beyond. Invited Speaker presented at Biologically-Inspired Robotics: The Legacy of W. Grey Walter.

Shannon, C. and Weaver, W. (1949). *The mathematical theory of communication*. University of Illinois Press.

Siegelmann, H. (1999). *Neural Networks and Analogue Computation: Beyond the Turing Limit*. Birkhäuser, Boston, MA.

Tegmark, M. (2000). The importance of quantum decoherence in brain processes. *Physical Review*, **61**, 4195–4206.

Turing, A. (1936). On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, **42**, 230–265.

Williamson, M. (1995). Series elastic actuators. Technical Report 1524, MIT AI Lab. AI Memo.

AISB Journal **1**(3), © SSAISB 2003
http://www.aisb.org.uk

On Learning by Exchanging Advice

Luís Nunes* and Eugénio Oliveira†

* ISCTE/LIACC-NIADR, FEUP Av. Dr. Roberto Frias 4200-465, Porto, Portugal, *Luis.Nunes@iscte.pt*
† FEUP/LIACC-NIADR, FEUP Av. Dr. Roberto Frias 4200-465, Porto, Portugal, *eco@fe.up.pt*

Abstract

One of the main questions concerning learning in Multi-Agent Systems is: "(How) can agents benefit from mutual interaction during the learning process?". This paper describes the study of an interactive advice-exchange mechanism as a possible way to improve agents' learning performance. The advice-exchange technique, discussed here, uses supervised learning (backpropagation), where reinforcement is not directly coming from the environment but is based on advice given by peers with better performance score (higher confidence), to enhance the performance of a heterogeneous group of Learning Agents (LAs). The LAs are facing similar problems, in an environment where only reinforcement information is available. Each LA applies a different, well known, learning technique: Random Walk, Simulated Annealing, Evolutionary Algorithms and Q-Learning. The problem used for evaluation is a simplified traffic-control simulation. In the following text the reader can find a description of the traffic simulation and Learning Agents (focused on the advice-exchange mechanism), a discussion of the first results obtained and suggested techniques to overcome the problems that have been observed. Initial results indicate that advice-exchange can improve learning speed, although "bad advice" and/or blind reliance can disturb the learning performance. The use of supervised learning to incorporate advice given from non-expert peers using different learning algorithms, in problems where no supervision information is available, is, to the best of the authors' knowledge, a new concept in the area of Multi-Agent Systems Learning.

1 Introduction

1.1 Framework

This work aims at contributing to answer the question: "(How) can agents benefit from mutual interaction during the learning process, in order to achieve better individual and overall system performances?". This question has been deemed a "challenging issue" by several authors in recently published work (Sen, 1996; Weiss and Dillenbourg, 1999; Kazakov and Kudenko, 2001; Mataric, 2001).

In the pursuit of an answer to this question, the objects of study are the interactions between the Learning Agents (hereafter referred to as agents for the sake of simplicity) and the effects these interactions have on individual and global learning processes. Interactions that affect the learning process can take several forms in Multi-Agent Systems (MAS). These forms range from the indirect effects of other agents' actions (whether

they are cooperative or competitive), to direct communication of complex knowledge structures, as well as cooperative negotiation of a search policy or solution.

The most promising way in which cooperative learning agents can benefit from interaction seems to be by exchanging (or sharing) information regarding the learning process itself. As observed by Tan (1993) agents can exchange information regarding several aspects of the learning process: a) the state of the environment, b) episodes (state, action, reward triplets), or c) internal parameters and policies.

Exchanging environment states can be seen as a form of shared exploration. Sharing this information may require a large amount of communication, although the use of a selective policy for the exchange of information may reduce this cost. This type of interaction may be seen as if each agent has extra sets of sensors spread out in the environment, being able to have a more complete view of its external state. This larger view of the state space may require either pre-acquired knowledge on how to interpret this information and integrate it with its own view of the environment's state, or simply be considered as extra input providing a wider range of information about the state. In the limit case, where all agents have access to information regarding the state sensed by all their peers, each agent could be seen as a classical Machine Learning (ML) system with distributed sensors if we consider other agents' actions as part of the environment. One interesting difference, though, is the fact that other agents sensors are not under the control of the learning agent and the perspective they provide on the world may be biased by the needs of the owner of the sensor.

Episode exchange requires that the agents are (or have been) facing similar problems, requiring similar solutions and may also lead to large amounts of communication if there is no criteria regulating the exchange of information. In the limit case, where all agents share all the episodes, this process can also be seen as a single learning system, and produce very little new knowledge. In fact, the exchange of too much data could lead all the agents to follow the same path through the search space, wasting valuable exploration resources.

Sharing internal parameters is another way in which agents can benefit from the knowledge obtained by their peers. Again, in the limit, this can be seen as the use of a single learning agent if communication is unrestricted. This type of information exchange requires that agents have similar internal structures, so that they can easily map their peers' internal parameters into their own, or that they share a complex domain ontology.

As can be seen in the above paragraphs the question is not only: "what type of information to exchange?" , but also "when to exchange information?" and "how much information to exchange?".

When considering human cooperative learning in a team, a common method to improve one's skills is to ask for advice at critical times, or request a demonstration of a solution to a particular problem from someone who is reputed to have better skills in the subject. This is what we have attempted to translate into the realm of Multi-Agent Systems Learning (MASL).

1.2 Rationale and summarized description

This paper reports experiments in which agents selectively share episodes by requesting advice for given situations to other agents whose score is, currently, better than their own in solving a particular problem. Considering the discussion of the previous section, this option seemed the most promising for the following reasons: a) Sharing episodes does not put heavy restrictions on the heterogeneity of the underlying learning algorithms,

as sharing of parameters does; b) Having different algorithms solving similar problems may lead to different forms of exploration of the same search space, thus increasing the probability of finding a good solution; c) It is more informative and less dependent on pre-coded knowledge than the exchange of environment's states.

Experiments were conducted with a group of agents embedded in a simplified simulation of a traffic control problem to test the advantages and problems of advice-exchange during learning. Each individual agent uses a standard version of a well-known, subsymbolic, learning algorithm (Random Walk, Evolutionary Algorithms, Simulated Annealing, and Q-Learning; see section 3.2.1 for details on these algorithms). Agents are heterogeneous (i.e., each applies a different learning mechanism, unknown to others). The information exchanged amongst agents is: current state (as seen by the advisee agent); best response that can be provided to that state (by the advisor agent); present and best scores, broadcasted at the end of each training stage (epoch). The problem chosen to test the use of advice-exchange has, as most problems studied in MASL, the following characteristics: a) Analytical computation of the optimal actions is intractable; b) The only information available to evaluate learning is a measure of the quality of the present state of the system; c) The same action executed by a given agent may have different consequences at different times, even if the system is (as far as the agent is allowed to know) in the same state; d) The agent has only a partial view of the problem's state.

The simplified traffic control problem chosen for these experiments requires that each agent learn to control the traffic-lights in one intersection under variable traffic conditions. Each intersection has four incoming, and four outgoing, lanes. One agent controls the four traffic lights necessary to discipline traffic in one intersection. In the experiments reported here, the crossings controlled by each of the agents are not connected. The learning parameters of each agent are adapted using two different methods: a reinforcement-based algorithm, using a quality measure that is directly supplied by the environment, and supervised learning using the advice given by peers as the desired response. Notice that the term "reinforcement-based" is used to mean "based on a scalar quality/utility feedback", as opposed to supervised learning which requires a desired response as feedback. The common usage of the term "reinforcement learning", that refers to variations of temporal difference methods (Sutton and Barto, 1987), is a subclass of reinforcement-based algorithms, as are, for instance, most flavours of Evolutionary Algorithms.

2 Related Work

The advantages and drawbacks of sharing information and using external teachers in variants of Q-Learning (Watkins and Dayan, 1992) had some important contributions in the early 90's. To situate the work presented below the remainder of this section will provide a review of the related work.

Whitehead (1991) reports on the usage of two cooperative learning mechanisms: Learning with an External Critic (LEC) and Learning By Watching (LBW). The first, (LEC), is based on the use of an external automated critic, while the second (LBW), learns vicariously by watching other agent's behaviour (which is equivalent to sharing series of: state, action, quality triplets). This work proves that the complexity of the search mechanisms of both LEC and LBW is inferior to that of standard Q-Learning for an important class of state-spaces. Experiments reported in (Whitehead and Ballard, 1991) support these conclusions.

Lin (1992) uses a human teacher to improve the performance of two variants of Q-Learning. This work reports that the *"advantages of teaching should become more rele-*

vant as the learning task gets more difficult", (Lin, 1992, section 6.4, page 315). Results presented show that teaching does improve learning performance in the harder task tested (a variant of the maze problem), although it seems to have no effect on the performance on the easier task (an easier variant of the same maze problem).

The main reference on related work is (Tan, 1993). Tan addressed the problem of exchanging information amongst Q-Learning agents during the learning process. This work reports the results of sharing several types of information amongst several (Q-Learning) agents in the predator-prey problem. Experiments were conducted in which agents shared policies, episodes, and sensation (state). Although the experiments use solely Q-Learning in the predator-prey domain, the author believes that: *"conclusions can be applied to cooperation among autonomous learning agents in general"*, (Tan, 1993, section 7, par 1). Conclusions point out that *"a) additional sensation from another agent is beneficial if it can be used efficiently, b) sharing learned policies or episodes among agents speeds up learning at the cost of communication, and c) for joint tasks, agents engaging in partnership can significantly outperform independent agents, although they may learn slowly in the beginning"*, (Tan, 1993, in abstract). The results presented in that paper also appear to point to the conclusion that sharing episodes with peers is beneficial and can lead to a performance similar to that obtained by sharing policies. Sharing episodes volunteered by an expert agent leads to the best scores in the presented tests, significantly outperforming all other agents in the experiments.

After these first works several variants of information sharing Q-Learners appeared reporting good results in the mixture of some form of teaching and reinforcement learning. The following paragraphs make a brief review of the recent work in this area.

Clouse (1996) uses an automatic expert trainer to give the agent actions to perform, thus reducing the exploration time.

Mataric (1996) reports on the use of localized communication to share sensory data and reward as a way to overcome hidden state and credit assignment problems in groups of agents. The experiments conducted in two robot problems, (block pushing and foraging) show improvements in performance on both cases. Later work by the same author, Mataric (2001) reports several good results using human teaching and learning by imitation in robot tasks.

Brafman and Tennemholtz (1996) use an expert agent to teach a student agent in a version of the "prisoner's dilemma". The agents implement variations of Q-Learning.

Maclin and Shavlik (1997) use human advice, encoded in rules, which are acquired in a programming language that was specially designed for this purpose. These rules are inserted in a Knowledge Based Neural Network (KBANN) used in Q-Learning to estimate the quality of a given action.

Berenji and Vengerov (2000) report analytical and experimental results concerning the cooperation of Q-Learning agents by sharing quality values amongst them. Experiments were conducted in two abstract problems. Results point out that limitations to cooperative learning described in (Whitehead, 1991) can be surpassed successfully under certain circumstances, leading to better results than the theoretical predictions foresaw.

Price and Boutilier (2000) use implicit imitation to accelerate reinforcement learning. The quality of the actions of each agent contains an extra term with information about the *"mentor's"* state-transition matrix. The *"student"* agent is induced into trying the actions that are more often chosen by the mentor.

Simultaneous uses of Evolutionary Algorithms (Holland, 1975; Koza, 1992) and Backpropagation (Rumelhart et al., 1986) are relatively common in Machine Learning (ML) literature, although in most cases Evolutionary Algorithms are used to select the topology or learning parameters, and not to update weights. Some examples can be found in

(Salustowicz, 1995) and (Yao, 1999). There are also reports on the successful use of Evolutionary Algorithms and Backpropagation simultaneously for weight adaptation (Topchy et al., 1996; Ku and Mak, 1997; Ehardh et al., 1998). Most of the problems in which a mixture of Evolutionary Algorithms and Backpropagation is used are supervised learning problems, i.e., problems for which the desired response of the system is known in advance (not the case of the problem studied in this paper).

Castillo et al. (1998) obtained good results in several standard ML problems using Simulated Annealing and Backpropagation, in a similar way to that which is applied in this work. Again, this was used as an add-on to supervised learning to solve a problem for which there is a well-known desired response.

The use of learning techniques for the control of traffic-lights can be found in (Goldman and Rosenschein, 1995; Thorpe, 1997; Brockfeld et al., 2001).

3 Experimental Setup

This section will describe the internal details of the traffic simulation, the learning mechanisms and the advice-exchange technique.

3.1 The Traffic Simulator

The traffic simulator environment is composed of lanes, lane-segments, traffic-lights (and the corresponding controlling agents), and cars.

Cars are "well behaved", in the sense that they: a) Can only move forward; b) Do not cross yellow or red-lights; c) Move at a constant speed; d) Do not crash into other cars.

Cars are inserted at the beginning of each lane, whenever that space is empty, with a probability that varies in time. The time-unit used throughout this description is one turn. One turn corresponds to a period where each object in the system is allowed to perform one action and all the necessary calculations for it. Each lane has three lane-segments: incoming (before the crossing, where cars are inserted), crossing and outgoing. Each local scenario consists of four lanes, each with a different movement direction and one crossing (the lanes in a local scenario will be referred as North, South, East and West, for the remainder of this description). In the experiments reported here the local scenarios are not connected, i.e., each lane has only one crossing and one traffic light. Cars are inserted in its incoming lane-segment and removed when they reach the extremity of its outgoing lane-segment, after having passed the crossing.

At the beginning of each green-yellow-red cycle, the agents observe the state of environment for their local scenario and decide on the percentage of green-time (gt) to attribute to the North and South lanes (the percentage of time attributed to the East and West lanes is automatically set to the remaining time. Yellow-time is fixed in each experiment and lies in the interval $[10, 15]$ turns).

Two types of description of the environment's state are used, the first is realistic in the sense that it is technically achievable to collect that type of data in a real situation and it is actually used by traffic controllers today. The second, although it may be unfeasible in today's traffic monitoring systems, was considered to have relevant information for the learning process. In the first type of state representation, the state, at a given time, is composed by four scalar values, where each component represents the ratio of the number of incoming vehicles in a given lane relative to the total number of incoming vehicles in all lanes. This state representation will be referred as *count state* representation.

The second type of environment state has the same information as the one described above plus four scalar values, each of which represents the lifetime (number of turns since creation) of the incoming vehicle that is closest to the traffic-light. To keep inputs within the interval [0,1], this value was cut-off at a maximum lifetime (*lifemax*), and divided by the same value. The value of *lifemax* was chosen to be 3 to 10 times the number of turns a car takes to reach the crossing at average speed, depending on the difficulty of each particular scenario, which is mainly dependent on the parameters used for car generation. This state representation will be referred as *count-time state* representation.

The state representations described above are similar to the ones that were reported to have produced some of the best results in the experiments conducted by Thorpe (1997) for the same type of problem (learning to control traffic-lights at an intersection).

The normalization of the inputs to fit the $[0, 1]$ interval was necessary, even at the cost of loss of information, because using percentages for the first four elements of the state space allows a substantial reduction of the number of possible states, as described below when the implementation of Q-Learning is discussed.

The quality of service of each traffic-light controller (Q), was initially calculated as a linear decreasing function of the average time cars used to cross the scenario (tc_i). This measure did not provide enough differentiation of "good" and "bad" environment states, thus a second function was introduced to emphasize the difference in quality between these two types of environment states. A comparative view of both functions can be seen in figure 1, the former in continuous line the latter in squares. The second function was created specifically to suit this particular scenario and different parameterizations were used in several trials. The shape of the function however was maintained in all trials, with a steep decrease at a given point to differentiate clearly between "good" and "bad" states.

The car generation parameters in traffic simulator proved difficult to tune. Slight changes led to simulations that were either too difficult (no heuristic nor any learned strategy were able to prevent major traffic jams), or to problems in which both simple heuristics and learned strategies were able to keep a normal traffic flow with very few learning steps.

The traffic simulator was coded in C++, with a Java graphical interface. Agents are not independent processes, at this stage, they are merely C++ objects that are given a turn to execute their actions in round-robin. On the one hand, this choice eliminates the "noise" of asynchronous communication and synchronization of parallel threads, on the other hand, lighter agents that perform simple but coarse learning techniques (like Random Walk) are being slowed down by the more computationally intensive learning algorithms (like Q-Learning).

Although this was not an issue, the simulation runs faster than real-time, even when all agents are performing learning steps. Simulations ran (usually) for 1600 epochs, where each epoch consists of 50 green-yellow-red cycles, each consisting of 100 turns in which, on average, approximately 150 cars were moved and checked for collisions. Each simulation, with five disconnected crossings (i.e., four parallel learning algorithms and one heuristic agent), took 4 to 5 hours to run in a Pentium IV at 1.5 GHz. To generate a set of comparable data, each scenario must be run twice: with and without advice-exchange.

3.2 Learning Agents

This section describes the learning algorithms used by each of the agents involved in the experiments, as well as the heuristic used for the fixed strategy agent.

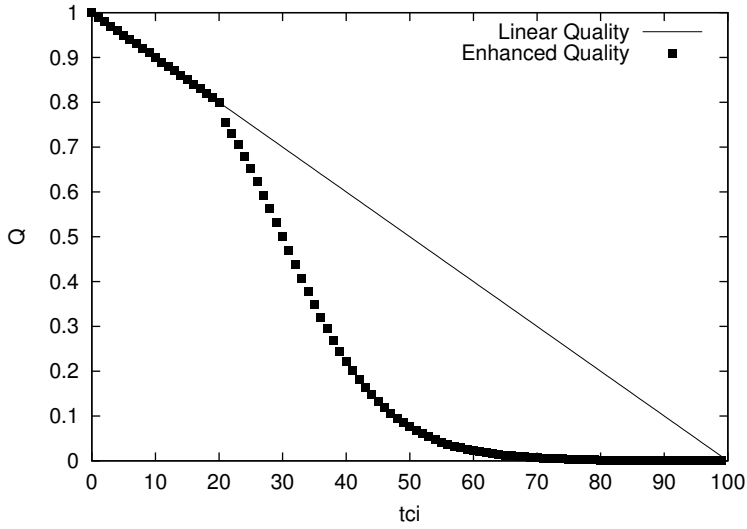

Figure 1: Two functions for the evaluation of traffic quality (Q) based on the average time of life of the incoming cars (tc_i).

3.2.1 Stand-alone agents

The stand-alone versions of the learning agents are used to provide results with which the performance of advice-exchanging agents could be compared. The stand-alone agents implement four classical learning algorithms: Random Walk (RW), Simulated Annealing (SA), Evolutionary Algorithms (EA) and Q-Learning (QL). A fifth agent was implemented (HEU) using a fixed heuristic policy.

As the objective of these experiments was not to solve this problem in the most efficient way, but to evaluate advice-exchange for problems that have characteristics similar to the ones stated above for the traffic-simulation problem, the algorithms were not chosen or fine-tuned to produce the best possible results for traffic control. The choice of algorithms and their parameters was guided by the goal of comparing the performance of a heterogeneous group of learning agents, using classical learning strategies, in a non-deterministic, non-supervised, partially-observable problem, with and without advice-exchange.

All agents, except QL and HEU, adapt the weights of a small, one hidden layer, neural network. Experiments were conducted with several topologies, but the results discussed below refer to fully connected networks of 4x4x1, when using *count state* representation, and 8x4x1, when using *count-time state* representation. The weights of these networks were initialised randomly with values in the range $[-0.5, 0.5]$. The hidden layer is composed of sigmoids whose output varies in $]-1, 1[$, while the outer layer sigmoids' output is in the range $]0, 1[$. The output will be the percentage of green-time (gt) for the North-South lane in the next green-yellow-red cycle.

The Random Walk (RW) algorithm simply disturbs the current values of the weights of the neural network by adding a random value in the range $[-d, d]$, where d is the maximum disturbance, which will be updated after a given number of epochs, (ne), according to $d_{i+1} = \lambda d_i$, with $0 < \lambda < 1$, until it reaches a minimum value, (d_{min}). An epoch consists of n green-red-yellow cycles. At the end of an epoch, the new set of parameters is kept if the average quality of service in the controlled crossing during that epoch

is better than the best average quality achieved so far. The values used for the parameters of this algorithm in the experiments discussed here were in the following intervals: $d \in [0.5, 0.7], d_{min} = 0.01, \lambda = 0.99, n = 50, ne \in [3, 7]$. These values apply also for the decay of disturbance limits in the following descriptions of SA and EA. When referring to the intervals in which values were chosen, it is meant that in different experiments several combinations of parameter values were tested but the initial value for these parameters was always in the mentioned range.

Simulated Annealing (SA), (Kirkpatrick et al., 1983), works in a similar way to Random Walk, but it may accept the new parameters even if the quality has diminished. New parameters are accepted if a uniformly generated random number $p \in [0, 1[$, is smaller than

$$pa(t) = e^{\frac{-\Delta q(t)}{T}}, \tag{1}$$

where T is a temperature parameter that is decreased during training in the same way as d in RW and Δq is the difference between the best average quality achieved so far and the average quality of the last epoch.

Evolutionary Algorithms (EA), (Holland, 1975; Koza, 1992), were implemented in a similar way to the one described in (Glickman and Sycara, 1999), which is reported to have been successful in learning to navigate in a difficult variation of the maze problem by updating the weights of a small Recurrent Artificial Neural Network. This implementation relies almost totally in the mutation of the weights, in a way similar to the one used for the disturbance of weights described for RW and SA. Each set of parameters (specimen), which comprises all the weights of a neural network of the appropriate size for the state representation being used, is evaluated during one epoch. After the whole population is evaluated, the best n specimens are chosen for mutation and recombination. An elitist strategy is used by keeping the best b specimens untouched for the next generation. The remainder of the population is built as follows: the first m are mutated, the remaining specimens (r) are created from pairs of the selected specimens, by choosing randomly from each of them entire layers of neural network weights. The values used for the parameters of this algorithm in the experiments discussed here were in the following intervals: $n \in [7, 10], b \in [3, 7], m \in [15, 25], r \in [2, 5]$. The size of the population was in $[20, 30]$.

Q-Learning (QL), (Watkins and Dayan, 1992), uses a lookup table with an entry for each state-action pair in which the expected utility $Q(s, a)$ is saved. $Q(s, a)$ represents the expected utility of doing action a when the environment is in state s. Utility is updated in the usual way, i.e.,

$$Q(s, a) = Q(s, a) + \alpha (r + \beta Q_{max}(s')), \tag{2}$$

where s' is the state after performing action a, α is the learning rate, β the discount factor and $Q_{max}(s')$ is given by

$$Q_{max}(s) = \max(Q(s, a)), \tag{3}$$

for all possible actions a when the system is in state s.

The values of α (learning rate), in the different experiments, were in the interval $[0.5, 0.7]$. The learning rate is updated after a given number of epochs, (ne), according to $\alpha_{i+1} = \lambda \alpha_i$, with $0 < \lambda < 1$, until it reaches a minimum value (which in this case was 0.012). In the experiments discussed here $ne = 5$. Parameter β (discount) was fixed in each experiment. Different values for β were tested within the interval $[0.6, 0.8]$. The choice of action a, given that the system is in state s, was done with probability $p(a|s)$

that is given by a Boltzman distribution,

$$p\left(a|s\right) = \frac{e^{\frac{Q\left(s,a\right)}{T}}}{\sum_{i \in A_s} e^{\frac{Q\left(s,a_i\right)}{T}}} \tag{4}$$

where T is a temperature parameter, whose initial value was in the interval $[0.3, 0.7]$ and was decayed in a similar way to the one described for α and A_s is the set of all actions available from state s. Since the state of the environment is a real-valued vector, a partition of the space in a square lattice is required to map environment states (continuous) to internal (discrete) states. The decision of which is the state of the environment at a given time is made by calculating the euclidean distance between the continuous valued world state and each of the discrete state representations and selecting the state with minimum distance. For the *count state* representation this partition consists in states composed of quadruples of the form: (x_1, x_2, x_3, x_4), for which $x_1 + x_2 + x_3 + x_4 = 1.0$, and $x_i \in \{0, 0.1, 0.2, ..., 0.9, 1.0\}$. This reduction of the state space, compared to the use of all possible quadruples with elements in $\{0.0, 0.1, 0.2, ...\}$, is possible given that the representation of the environment is composed of the percentages of vehicles in each lane relative to the number of vehicles in all lanes, thus being restricted to quadruples for which the sum of all elements is 1.0. For the count-time state representation the internal state is of the form: $(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8)$, where the first four parameters are generated in the same fashion as in the previous case but with a coarser granularity and, the last four elements, are selected combinations of values in $\{0.0, 0.25, 0.5, 0.75, 1.0\}$. The number of states for the first and second case is, respectively, 286 and 1225.

Actions, i.e., green-time for the North and South lanes, are also considered as discrete values starting from zero, up to the maximum green time allowed, and differing by 0.05 steps.

The heuristic agent (HEU) gives a response that is calculated in different ways, depending on the state representation. The percentage of green-time is proportional to the number of cars in the North-South lanes, relative to the total number of cars for the *count state* representation, and in a similar way accounting for the lifetime values for the first car in each track for the *count-time state* representation.

3.2.2 Advice-exchange mechanism

The main expectation, when advice-exchange was chosen, was that using advice from the more knowledgeable agents in the system would improve the learning performances of all agents. Since supervision is a more efficient training method than reinforcement, (at the expense of needing more information) then, when no supervision information is available it seems reasonable to use advice as supervision. Better yet, if agents have different learning skills, which produce different types of progress through the search-space, they may be able to avoid that others get stuck in local minima by exchanging advice. It is unlikely that all agents are stuck in the same local minima and the exchange of information regarding the appropriate answers to some environment states could force others to seek better solutions.

The process of advice-exchange is conducted in a different way in the agents that use a neural network as activation function and in the Q-Learning agent. The heuristic agent does not participate in the experiments concerning advice-exchange. Advice-exchange is prohibited in the first 2 to 10 epochs of training, depending on the experiments, to avoid random advice being exchanged and to allow some time for the agents to announce a credible best average quality value.

1. Agent i: receive the best average quality (bq_j) from all
other agents ($j \neq i$). Quality for Agent i is cq_i.
i: get state s for evaluation.
$arg\_max_j \left(bq_j\right)$, for all agents ($j \neq i$).
4. Agent i: if $cq_i < d \max \left(bq_j\right)$:
   a. Agent i: send agent k the current state s and request advice.
   b. Agent k: switch to best parameters and run state s
to produce its best guess at the adequate response (gt).
   c. Agent k: return gt to Agent i.
   d. Agent i: process advice (gt).
5. Agent i: run state s and produce response gt'.

Table 1: Steps of the advice-exchange sequence for an advisee agent (i) and an advisor agent (k).

All agents broadcast their best result (i.e., best average quality measured during one epoch) at the beginning of each epoch. At the beginning of each green-yellow-red cycle, agent i (the advisee) evaluates its current average quality (cq_i) since the beginning of the present epoch. This quality is compared with the best average quality (bq_j), for all agents j. Let $mbq_k = \max \left(bq_j\right)$, for all agents $j \neq i$. If $cq_i < d\, mbq_k$ where d is a discount factor (usually 0.8), then agent i will request advice from agent k (the advisor) who as advertised the best average quality. The request for advice is sent having as parameter the current state of the environment as seen by agent i. The advisor switches his working parameters (neural network weights in most cases) to the set of parameters that was used in the epoch where the best average quality was achieved and runs the state communicated by the advisee producing its best guess at what would be the appropriate response to this state. This response (the advised percentage of green time for the north and south lanes) is communicated back to the advisee. In the case where advisees are RW, SA and EA agents, the communicated result is used as desired response for on-line backpropagation (Rumelhart et al., 1986) applied to the weights of the neural-network. In some experiments an adaptive learning rate backpropagation algorithm (Silva and Almeida, 1990) was used but results were not significantly different. The values for the main backpropagation parameters used in the experiments discussed here were in the following intervals: learning rate $\in [0.001, 0.05]$, momentum $\in [0.3, 0.7]$.

When the Q-Learning agent is the advisor, switching to best parameters corresponds simply to selecting the action with best quality. In the case where the Q-Learning agent is the advisee, the action that is closest to the given advice (recall that actions are discrete values in this case) is rewarded in a similar way to that described in (2). Since in this case the state of the system after action a is unknown, the value of $Q_{max}\left(s'\right)$ is replaced by a weighted average of the utilities of all the possible following states when executing action a at state s:

$$Q_{max}\left(a, s\right) = \sum_{s' \in S_{sa}} p\left(s'|a, s\right) Q_{max}\left(s'\right) \tag{5}$$

where $p\left(s'|a, s\right)$ is the probability of a transition to state s' given that action a is executed at state s and it is calculated based on previous experience, as the ratio between the number of transitions ($nt_{s'as}$) to state s' when performing action a at the current state, s, relative

to the total number of transitions from current state by action a, i.e.,

$$p\left(s'|a,s\right) = \frac{nt_{s'as}}{\sum_i nt_{ias}}, i \in S_{sa}, \tag{6}$$

where S_{sa} is the set of states reachable from state s by action a. This type of adaptation of the state utility was proposed in (Sutton, 1992). After updating the internal parameters with the advised information, the advisee agent gives the appropriate response to the system following the normal procedure for each particular algorithm.

4 Experimental Results

Before the discussion of the experimental results, let us put forward a few brief remarks concerning the simulation and experiments. The type of problem delt with is a difficult topic for simulation. Several works have been done in this area, and the simplifications made in this scenario were, in great measure, inspired by previous works mentioned in section 2.

Quite frequently the car-generation parameters created problems that tended to be either too easy or too hard, and, in the first experiments, only marginal differences could be observed in the quality measure during training. The most interesting experiments conducted were the cases where lanes had quite different behaviours from one another, however, there seems to be a fine line between hard solvable problems and, apparently, insoluble tasks in which no learning strategy, nor heuristic, could reach reasonable values of quality.

The interpretation of results has also raised some problems. The fact that agents are running online, and most of them are based on random disturbance, added to the stochastic nature of the environment, produces very "noisy" quality evaluations. The results presented here focus mainly on the analysis of the evolution of the best quality achieved up the present moment of training. Other measures also give us an insight on the process, but, at the present moment, and given the space limitations, this seemed to be the one that could better illustrate the main observations made during experiments.

The above-mentioned stochastic nature of the problem, and the large simulation times, also forced a compromise in the choice of parameters for car generation. Although a greater variety of behaviours could be achieved with other type of functions, whose periods span over a larger time-frame, this would require that each training epoch would be much longer, so that a comparison between values of different epochs would be fair.

One last remark concerning the discussion of results that will follow. The amount data necessary for a sound statistical comparison and evaluation of this technique is still being gathered. The preliminary results discussed here, produced in a series of 30 full trials, give us an insight on the problems and possible advantages of advice-exchange during learning, but data is still not sufficient for a detailed evaluation of the advantages and drawbacks of this technique. The above mentioned trials were run under different conditions, either in the parameters of car-generation, lane-size and car speeds, or in the parameters of the algorithms themselves.

Before starting experiments, some results were expected, namely:

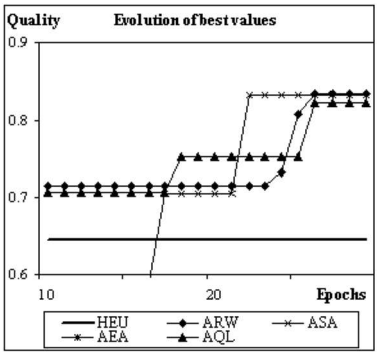

Figure 2: Detail of the initial phase of a trial where advice given by Simulated Annealing (ASA) led Random Walk (ARW) and Q-Learning (AQL) agents on a sudden climb of more than 10%. Evolutionary Algorithms also benefited from this jump, but the climb was less steep and from a lower point.

a) Initial disturbance of the learning process due to advice by non-expert peers, as reported by Tan (1993) for cooperation amongst Q-Learning agents.
b) In the initial phase of training, fast, step-like, increases in quality of response, as soon as one of the agents, found a better area of the state space and drove other agents that had poorer performances to that area.
c) Final convergence on better quality values than in tests where no advice is exchanged.
d) Problems of convergence when using excess of advice, or high learning rates when processing advice.
e) Improved resistance to bad initial parameters.

The actual observed results differed in some respects from expectations. The initial disturbance, or slower convergence, reported by Tan (1993) for Q-Learning agents, was not observed as a rule, although it occasionally happened. The exact opposite was observed more frequently, which seems indicate that this is an advantage that is particular to heterogeneous groups of learning agents. In some experiments we can find agents that use advice climbing much faster to a reasonable quality plateau. Occasionally learning was much slower afterwards (probably a local maximum was reached) and this high initial quality value was gradually surpassed by the stand-alone algorithms during the rest of the training. The second expectation, the appearance of high steps in the quality measure, due to advice from an agent that discovered a much better area of the search-space, was observed, but seems to be less common than expected. Figure 2 shows a detail of the initial phase of a trial where we can see a typical situation of the described behaviour. The Simulated Annealing agent jumps to a high quality area, and "pulls" Random Walk and Q-Learning into that area in a few epochs. In this experiment the advice-exchanging algorithms did not stop at this quality plateau, being able to obtain better scores than their counterparts.

Results where the final quality values for the best agent, on trials with advice-exchange, is significantly better than in the normal case were observed, but do not seem to be as common as expected. Figures 3 and 4 show comparisons of the methods with and without advice-exchange for one of the trials where advice-exchange proved advantageous. Notice that all results are better than the one obtained by the heuristic agent (HEU), which

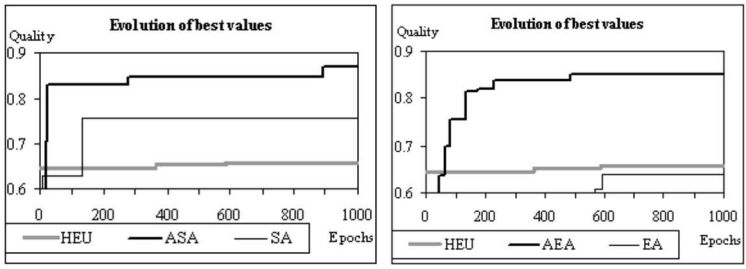

Figure 3: Left: Comparison of Simulated Annealing performance, with (ASA) and without (SA) advice-exchange, and the corresponding heuristic (HEU) quality for the same trial. Right: Comparison of Evolutionary Algorithms performance, with (AEA) and without (EA) advice-exchange, and the corresponding heuristic (HEU) quality for the same trial.

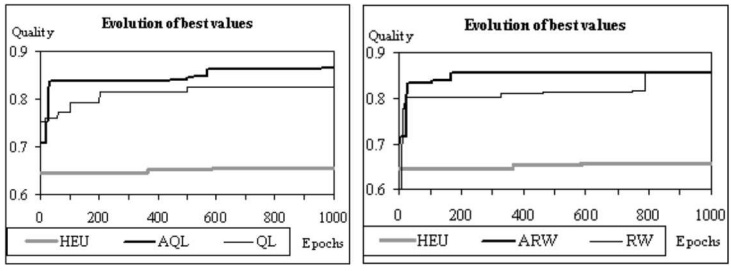

Figure 4: Left: Comparison of Q-Learning performance, with (AQL) and without (QL) advice-exchange, and the corresponding heuristic (HEU) quality for the same trial. Right: Comparison of Random Walk performance, with (ARW) and without (RW) advice-exchange, and the corresponding heuristic (HEU) quality for the same trial

was not frequent. The most usual result is that agents climb to the vicinity of the best agent's quality in few epochs, learning to achieve a reasonably good result much faster than when not exchanging advice.

The expectations referred in d) and e) were observed, as was foreseen. In fact, several cases were observed in trials without advice-exchange, where early freezing of the temperature parameter or the decay of the exploration rate, led to a sudden stop at a low local quality maximum, from which the algorithm did not escape for the rest of trial. These events are rare in trials using advice-exchange.

One of the most interesting problems observed was that of ill advice. It was observed that some agents, due to a "lucky" initialisation and exploration sequence, never experience very heavy traffic conditions, thus, their best parameters are not suited to deal with this problem. When asked for advice regarding a heavy traffic situation, their advice is not only useless, but harmful, because it is stamped with the "quality" of an expert. In Q-Learning this was easy to observe because there were situations, far into the trials, for which advice was being given concerning states that had never been visited before. In the next section some measures to prevent this problem will be discussed.

5 Conclusions and Future Work

As mentioned in the previous section, advice-exchange seems to be a promising way in which agents can profit from mutual interaction during the learning process. However, this is just the beginning of a search, where a few questions were answered and many were raised. A thorough analysis of the conditions in which this technique is advantageous is still necessary. It is important to discover how this technique performs when agents are not just communicating information about similar learning problems, but attempting to solve the same problem in a common environment. The application of similar methods to other type of learning agents, as well as other problems, is also an important step in the validation of this approach.

For the time being, a more realistic traffic environment is under development based on the Nagel-Schreckenberg model for traffic simulation (Nagel and Shreckenberg, 1992). We hope that this new formulation provides a richer environment in which advice-exchange can be more thoroughly tested. One of the main problems observed with advice-exchange is that bad advice, or blind reliance, can hinder the learning process, sometimes beyond recovery. One of the major hopes to deal with this problem is to develop a technique in which advisors can measure the quality of their own advice, and advisees can develop trust relationships, which would provide a way to filter bad advice. This may be especially interesting if trust can be associated with agent-situation pairs, and may allow the advisee to differentiate who is the expert on the particular situation it is facing. Work on "trust" has been reported recently in several publications, one of the most interesting for the related subject being (Sen et al., 2000).

Another interesting issue rises from the fact that humans usually offer unrequested advice for limit situations. Either great new discoveries or actions that may be harmful for the advisee seem to be of paramount importance in the use of advice. Rendering unrequested advice at critical points, by showing episodes of limit situations, also seems like a promising approach to improve the skills of a group of learning agents. The same applies to the combination of advice from several sources.

These techniques may require an extra level of skills: more elaborate communication and planning capabilities, long-term memory, etc. These capabilities fall more into the realm of symbolic systems. The connection between symbolic and sub-symbolic layers, which has been also an interesting and rich topic of research in recent years, may play an important role in taking full advantage of some of the concepts outlined in this work. Our major aim is to, through a set of experiments, derive some principles and laws under which learning in the Multi Agent System framework proves to be more effective, and inherently different from just having agents learning as individuals (even if they are interacting in the same environment).

Acknowledgements

The authors would like to thank, Manuel Sequeira, Thibault Langlois, Jorge Louçã , Pedro Figueiredo, Ana Violante, Rui Lopes, Ricardo Ribeiro, Francisco Pires, Isabel Machado, Sofia Regojo and two anonymous reviewers. Also, our thanks to the ResearchIndex crew.

References

Berenji, H. R. and Vengerov, D. (2000). Advantages of cooperation between reinforcement learning agents in difficult stochastic problems. *9th IEEE International Confer-*

ence on Fuzzy Systems (FUZZ-IEEE '00).

Brafman, R. I. and Tennemholtz, M. (1996). On partially controlled multi-agent systems. *Journal of Artificial Intelligence Research*, (4):477–507.

Brockfeld, E., Barlovic, R., Shadshneider, A., and Shreckenberg, M. (2001). Optimizing traffic lights in a cellular automaton model for city traffic. *Physical Review*, (64).

Castillo, P. A., Gonzalez, J., Merelo, J. J., V. Rivas, G. R., and Prieto, A. (1998). Saprop: Optimization of multilayer perceptron parameters using simulated annealing. *IWANN99*.

Clouse, J. A. (1996). *Learning from an automated training agent*. Springer Verlag, Berlin.

Ehardh, W., Fink, T., Gutzman, M. M., Rahn, C., and Galicki, A. D. M. (1998). The improvement and comparison of different algorithms for optimizing neural networks on the maspar MP-2. *Neural Computation '98*, pages 617–623.

Glickman, M. and Sycara, K. (1999). Evolution of goal-directed behavior using limited information in a complex environment. *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference*.

Goldman, C. and Rosenschein, J. (1995). Mutually supervised learning in multi-agent systems. *Proceedings of the IJCAI-95 Workshop on Adaptation and Learning in Multi-Agent Systems, Motreal, CA*.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press.

Kazakov, D. and Kudenko, D. (2001). Machine learning and inductive logic programmimng for multi-agent systems. *Multi Agents Systems and Applications: 9th EACCAI advanced course, Selected Tutorial Papers*, pages 246–271.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220:671–680.

Koza, J. R. (1992). *Genetic programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge MA.

Ku, K. W. C. and Mak, M. W. (1997). Exploring the effects of lamarckian and baldwinian learning in evolving recurrent neural networks. *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 617–621.

Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8:293–321.

Maclin, R. and Shavlik, J. (1997). Creating advicetaking reinforcement learners. *Machine Learning*, 22:251–281.

Mataric, M. J. (1996). Using communication to reduce locality in distributed multi-agent learning. Computer Science Technical Report CS-96-190, Brandeis University.

Mataric, M. J. (2001). Learning in behaviour-based multi-robot sysytems: policies, models and other agents. *Journal of Cognitive Systems Research*, (2):81–93.

Nagel, K. and Shreckenberg, M. (1992). A cellular automaton model for freeway traffic. *J. Phisique I*, 2(12):2221–2229.

Price, B. and Boutilier, C. (2000). Imitation and reinforcement learning in agents with heterogeneous actions. *Seveteenth International Conference on Machine Learning ICML2000*.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. *Parallel Distributed Processing: Exploration in the Microstructure of Cognition*, 1:318–362.

Salustowicz, R. (1995). *A Genetic Algorithm for the Topological Optimization of Neural Networks*. PhD thesis, Tech. Univ. Berlin.

Sen, S. (1996). Reciprocity: a foundational principle for promoting cooperative behavior among self-interested agents. *Proc. of the Second International Conference on Multiagent Systems*, pages 322–329.

Sen, S., Biswas, A., and Debnath, S. (2000). Believing others: Pros and cons. *Proceedings of the Fourth International Conference on Multiagent Systems*, pages 279–286.

Silva, F. M. and Almeida, L. B. (1990). Speeding up backpropagation. *Advanced Neural Computers*, pages 151–158.

Sutton, R. S. (1992). Reinforcement learning architectures. *Proceedings ISKIT'92 International Symposium on Neural Information Processing*.

Sutton, R. S. and Barto, A. G. (1987). A temporal-difference model of classical conditioning. Technical Report TR87-509.2, GTE Labs.

Tan, M. (1993). Multi-agent reinforcement learning: Independent vs. cooperative agents. *Proceedings of the Tenth International Conference on Machine Learning*, pages 330–337.

Thorpe, T. (1997). Multi-agent reinforcement learning: Independent vs. cooperative agents. Master's thesis, Department of Computer Science, Colorado State University.

Topchy, A., Lebedko, O., and Miagkikh, V. (1996). Fast learning in multilayered neural networks by means of hybrid evolutionary and gradient algorithms. *Proc. of IC on Evolutionary Computation and Its Applications*.

Watkins, C. J. C. H. and Dayan, P. D. (1992). Technical note: Q-learning. *Machine Learning*, 8(3):279–292.

Weiss, G. and Dillenbourg, P. (1999). *What is 'multi' in multiagent learning*, chapter 4, pages 64–80. Pergamon Press.

Whitehead, S. D. (1991). A complexity analisys of cooperative mechanisms in reinforcement learning. *Proc. of the 9th National Conference on Artificial Inteligence (AAAI-91)*, pages 607–613.

Whitehead, S. D. and Ballard, D. H. (1991). A study of cooperative mechanisms for faster reinforcement learning. Technical Report TR 365, Computer Science Department, University of Rochester.

Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE, 87(9)*, pages 1423–1447.

AISB Journal **1**(3), © SSAISB, 2003
http://www.aisb.org.uk

The horse-bird creature generation experiment

Francisco C. Pereira and Amílcar Cardoso

Centro de Informática e Sistemas da Universidade de Coimbra (CISUC)
Dep. Engenharia Informática, Pólo II, Pinhal de Marrocos, 3030 Coimbra
camara@dei.uc.pt ; amilcar@dei.uc.pt

Abstract

This paper presents the process and results of experiments regarding the generation of blends of a concept of "horse" with a concept of "bird". The blending process is based on the framework of Conceptual Blending (Fauconnier and Turner, 1998) and its development is achieving some stability. We present an overview of our system, Divago, namely of its newest developments around the optimality constraints.
The results demonstrate the behavior of the system with regard to each of the optimality constraints and also give an insight on its ability for the generation of new concepts from the combination of pre-existing ones, although highlighting problems and further developments that must be taken.

1 Introduction

One big challenge to AI, more specifically to Computational Creativity, is that of the generation of new concepts. The first issue to approach is the very definition of *concept* and its representation, interesting issues on their own right. Assuming a concept representation and semantics, we are then faced with the problem of the *process*. What kind of processes can yield new and valid concepts?
In this paper, we apply a model of creative process that follows a framework, named Conceptual Blending (CB) (Fauconnier and Turner, 1998), and present some results of recent experiments. Although lacking in formalization and scientific proof in some aspects, this framework suggests principles and processes that explain many creative cognitive phenomena, such as metaphor, analogy and conceptual combination. CB is, at the least, a very elegant model of creativity, a motivation that led us to attempt to a computational basis. In the system we are developing, *Divago*, those principles and processes are applied iteratively until a *stable* solution is found. This solution should be a *blend*, a new concept (or web of concepts) that shares structure and knowledge from the inputs, yet having an emerging structure of its own (e.g. a "pegasus", as a blend of "horse" and "bird").
We start this paper by a short review of similar systems, namely from a related area named Conceptual Combination, after which we give an overview of the CB framework. Divago is presented afterwards and, finally, we present and analyse the experiments we made with the "horse" and "bird" domains, which constitutes the main contribution of this paper. The reader will also find a final discussion, in which we make a reflection around the results, the presented model and its creative aspects.

2 State of the Art

The first computational work on Conceptual Combination we find in literature is that of Carl Andersen (Andersen, 1996), which presents a system for "joining of information from two existing concepts to form a third, more complex concept". He gives a set of very interesting ideas, but the paper is lacking argumentation and validation, thus oversimplifying conceptual combination. An example of combination of "house" and "boat" is given, but the definition of these two initial concepts, from our point of view, biases the results because of their overt simplicity. Another issue is the lack of *background knowledge*, i.e., each concept is considered in isolation (a fact the author himself acknowledges), so there are no ontological explanations or means of relating the concepts in question other than from their structure, leaving to an external entity the task of establishing a mapping between them.

Fintan Costello and Mark Keane (Costello and Keane, 2000) bring us a computational model, C^3, for the interpretation of noun-noun compounds (e.g. "Cactus fish", "pet shark"), proposing one or more solutions for each concept pairing and validating them against empirical tests on people. C^3 searches for concept explanations that use differentiating properties from each of the nouns (the *diagnosticity* constraint), that are consistent with background knowledge (the *plausibility* constraint) and that avoid redundancy or vagueness (the *informativeness* constraint). In so doing, their system provides different sorts of noun-noun combinations, thus resulting in the polysemy we also find in humans. Noun-noun compounds are clearly one example of the conceptual combination and creativity we do regularly.

On the side of Conceptual Blending, Tony Veale and Diarmuid O'Donogue (Veale and O'Donogue, 2000) describe, from a computational perspective, a proposal inspired by Veale's Metaphor interpretation framework, *Sapper*. As we argue in (Pereira and Cardoso, 2001), this proposal lacks some fundamental points of CB, namely the emergence of a new domain, the blend, independently of the initial inputs. Furthermore, it takes into account only *metaphoric blends*.

Our system, Divago, initially proposed in (Pereira, 1998), formalized in (Pereira and Cardoso, 2001; **?**; Pereira and Cardoso, 2003)), makes use of a computational version of Conceptual Blending as a process for transformation of the search space. This motivation was discussed in (Pereira and Cardoso, 2002b), and the first experiments with blending a "house" and a "boat" are shown in (Pereira and Cardoso, 2002a) and demonstrate a change to the search space (e.g. containing houses with circular windows). Divago has a knowledge base composed of domains, instances and rules and blends them following eight optimality principles (described below). It makes use of a *generic domain* to find mappings between concepts, generic frames and rules and integrity constraints. It is expected to do concept combination as in (Andersen, 1996) and make noun-noun compound interpretations as in (Costello and Keane, 2000). The experiments shown in this paper focus on the former, while the ones presented in (Pereira, 2003) concerned to the latter.

3 Conceptual Blending

Conceptual Blending was initially proposed by (Fauconnier and Turner, 1998) as part of a major framework concerning cognition and language. Its role was to explain the integration of knowledge coming from distinct sources into a single, independent and coherent unit, the Blend. A blend is a concept or web of concepts whose existence and identity, al-

though attached to the pieces of knowledge that participated in its generation (the inputs), acquires gradual independence through use.

We find examples of blends in many sorts of situations. A blend can be an effective way to get attention and curiosity towards advertising a product (e.g. Sony's AIBO robot uses all sorts of Sony products behaving as if it were a real human) or spreading a message (e.g. the Marlboro cowboy with impotence problems). People have been making blends at least from the times of Greek mythology (e.g. *pegasus*) till today (e.g. *pokemons*) and is present throughout our daily communication (e.g. "John digested the book", "Sue sneezed the napkin off the table"). Many more examples and situations could be listed and studied in detail, demonstrating the ubiquity of CB.

In the *canonical* model of Conceptual Blending, we have four different *spaces*: two input spaces, one generic space and the blend. Each space corresponds to what Fauconnier and Turner call a "mental space", a cognitive structure that corresponds to a concept, a set of concepts, a frame, a reasoning or *lower level* entities like the perception of movement or the feeling of physical pain. Mental spaces may have internal connections (inner-space relations) between their constituent elements and connections to other mental spaces (outer-space relations). The input spaces correspond to two mental spaces (e.g. horse and bird) that will be integrated in the blend (e.g. a horse with wings). The generic space contains knowledge that is not specific to any of the inputs but may relate to both (e.g. biology taxonomies) or is common sense (e.g. Greek mythology).

An essential step in the process of blending is the establishment of a (partial) mapping between elements of the input spaces. This mapping may be achieved through different processes (e.g. identity, structure alignment, slot-filling, analogy) and doesn't have to be 1-to-1. The paired elements are projected onto the blend as well as other surrounding elements and relations. This is a *selective projection*, i.e., some elements get projected to the blend, some don't.

From the projections, some new relations emerge that relate elements either as a direct result from the projection or from "running the blend", which consists of performing cognitive work within the blend, according to its own emergent logic. There is a set of *governing principles*, the *Optimality Pressures*, that should drive the process of generating a "good blend" (Fauconnier and Turner, 1998):

- Integration - The blend must constitute a tightly integrated scene that can be manipulated as a unit. More generally, every space in the blend structure should have integration.

- Pattern Completion - Other things being equal, complete elements in the blend by using existing integrated patterns as additional inputs. Other things being equal, use a completing frame that has relations that can be the compressed versions of the important outer-space vital relations between the inputs.

- Topology - For any input space and any element in that space projected into the blend, it is optimal for the relations of the element in the blend to match the relations of its counterpart.

- Maximization of Vital Relations - Other things being equal, maximize the vital relations in the network. In particular, maximize the vital relations in the blended space and reflect them in outer-space vital relations.[1]

[1] Fauconnier and Turner identify 15 different vital relations: change, identity, time, space, cause-effect, part-whole, representation, role, analogy, disanalogy, property, similarity, category, intentionality and uniqueness

- Intensification of Vital Relations - Other things being equal, intensify vital relations.
- Unpacking - The blend alone must enable the understander to unpack the blend to reconstruct the inputs, the cross-space mapping, the generic space, and the network of connections between all these spaces
- Web - Manipulating the blend as a unit must maintain the web of appropriate connections to the input spaces easily and without additional surveillance or computation.
- Relevance - Other things being equal, an element in the blend should have relevance, including relevance for establishing links to other spaces and for running the blend. Conversely, an outer-space relation between the inputs that is important for the purpose of the network should have a corresponding compression in the blend.

These constraints work as *competing pressures* and their individual influence in the process should vary according to the situation; when the value of one grows, others decrease. As far as we know, there is no work yet towards an objective study of the optimality pressures, measuring examples of blends or specifying these principles in detail. This, we believe, inhibits considerably the appreciation and application of Conceptual Blending in scientific research, making a particular motivation for this work being that of testing and specifying a formal proposal of these optimality pressures.

4 Overview of the Model

The architecture of Divago has four central modules: The Knowledge Base, the Mapper, the Factory and the Constraints module. The Knowledge Base comprises data structures (the concept maps, the frames, the integrity constraints and the instances) that are organised in *spaces*, according to their scope and generality. The Mapper establishes mappings between the *input* spaces (e.g. some concepts from the input space “horse” get mapped onto concepts from the input space “bird”), and provides these associations to the Factory, which deals with the process of blend generation, controlled through the evaluation made by the Constraints module. This module implements the optimality constraints which, when together applied to a given blend, return a value between 0 and 1. A fifth module, Elaboration, will be added to the final version of the system and will transform the blend according to knowledge from the generic space (by applying rules and triggering frame conclusions).

4.1 The Knowledge Base

As in any other AI system, knowledge representation is the first fundamental issue to decide. Here, we are particularly concerned about the representation of a “concept”, for it is the goal of Divago to generate new “concepts”. Assuming a symbolic approach (as opposed to sub-symbolic ones, such as neural networks or genetic algorithms), we decided for a semantic network based representation, in which a concept does not stand alone as an isolated symbol, its definition and explanation being dependent on the relationships it has with the surrounding concepts. More specifically, we take our concept networks as being *Concept Maps*. A Concept Map is a graph in which nodes represent *concepts* and arcs represent *relations*. This is far from a novel perspective. It goes in consonance with Murphy and Medin’s mini-theories (Murphy and Medin, 1985) or CYC (Lenat, 1995)

and WordNet (Miller, 1995) representations. Even more important, any of the previously mentioned works (Costello and Keane, 2000; Andersen, 1996; Veale, 1997) suggest this view of concepts.

From a semiotics perspective, this representation of concepts seems Saussurian because "everything depends on relations"(de Saussure, 1983). We try to escape from this extreme position through the possibility of association of effective semantics to each concept (e.g. the concept "window" may be realized as a set of instructions for "drawing a square") and by the association of the concepts to practical examples, the *instances* (as in (Pereira and Cardoso, 2002a)). Now, from a Peircian point of view, imagining the *meaning triangle* (Ogden and Richards, 1923), we have the individual *symbol* (e.g. the word "window") as standing for a *concept* (e.g. the concept network around "window") and corresponding to an *object* (e.g. a drawing of a window).

The choice of symbols for concepts and relations in our concept maps is arbitrary, yet we are following a set of normalization principles. The first one is that relations must belong to the Generalized Upper Model hierarchy (GUM) (J. Bateman and Fabris, 1995), a general task and domain independent *linguistically motivated ontology* that intends to significantly simplify the interface between domain-specific knowledge and general linguistic resources. GUM occupies a level of abstraction midway between surface linguistic realizations and *conceptual* or *contextual* representations. Being split into two hierarchies, one containing all the concepts and the other all the roles, GUM gives us a large set of primitive relations to standardize our choices in the concept map. It is important to notice that, in our maps, the members of the concept hierarchy of GUM (e.g. "color", "ability", etc.) are also used as relations (e.g. "color(mane, dark)", "ability(horse, run)"). Other principles we follow in the construction of the concept maps is that concepts in our knowledge base may only be represented as nouns, adjectives, preferably in the singular form, or numerals (in the particular case of numbers). As we said, these are only normalization principles for the construction of the concept maps, so, in theory, the model itself doesn't take into account the lexical categories of the words used, following only the principle that "the same word corresponds to the same concept". In Divago, there are also other elements (such as instances), but for the scope of this paper, the reader needs only to understand the notion of *concept map*.

In Table 2, we show examples of concept maps of "horse" and "bird". These maps are necessarily arbitrary in the sense that each person would draw their own maps, a result of the different conceptualization and points of view one can take individually. Yet, we assume these as being the conceptualization of the domains of "horse" and "bird" and so, when we interpret a new concept as being a "bird with a moustache", we refer to that specific "bird" concept map with an attached subgraph that represents a "moustache".

The semantics of each individual relation is arbitrarily defined by the user. By default, each relation is simply a symbolic connection between two concepts (e.g. in "purpose(leg, stand)", "purpose" is a connection between "leg" and "stand"), being its interpretation dependent on a context (e.g. in (Pereira and Cardoso, 2002a), the "shape" relation and its parameters were converted into a set of drawing primitives). Structurally, the user is allowed to add *integrity constraints* to the relations (e.g. a "pw", or part-whole, relation cannot be circular, i.e. "pw(X,X)" is not possible) as well as to attach it to the GUM hierarchy (e.g. "eat", not in the original GUM, descends from "dispositive material action"). There is a particular relation that has a special role, "isa", which attaches a concept to a taxonomic hierarchy (the general ontology) that is included in Divago Knowledge Base.

Two other important knowledge structures to refer here are the *frames* and the *integrity constraints*. The frames have the role of describing specific composite concepts, situations

isa(horse,equinae)	pw(leg, horse)	purpose(horse, food)
isa(equinae,mammal)	purpose(leg, stand)	sound(horse, neigh)
existence(horse, farm)	pw(paw, leg)	purpose(mouth, eat)
existence(horse, wilderness)	purpose(horse, traction)	purpose(ear, hear)
pw(snout, horse)	eat(horse, grass)	color(mane, dark)
pw(mane, horse)	ability(horse, run)	size(mane, long)
pw(tail, horse)	carrier(horse, human)	material(mane, hair)
quantity(paw, 4)	quantity(leg, 4)	purpose(horse, cargo)
pw(eye, snout)	quantity(eye, 2)	taxonomicq(horse, ruminant)
pw(ear, snout)	quantity(ear, 2)	ride(human, horse)
pw(mouth,snout)	purpose(eye, see)	motion_process(horse,walk)
isa(farm, human_setting)		

Table 1: The concept map of *horse*

isa(bird, aves)	existence(bird, house)	isa(aves,oviparous)
lay(oviparous, egg)	existence(bird,wilderness)	purpose(bird, pet)
purpose(bird, food)	purpose(eye, see)	smaller_than(bird, human)
pw(lung, bird)	motion_process(bird, fly)	purpose(beak, chirp)
purpose(lung, breathe)	quantity(eye, 2)	quantity(wing, 2)
isa(owl, bird)	isa(paradise_bird, bird)	quantity(claw, 2)
ability(bird, fly)	pw(wing, bird)	conditional(wing, fly)
pw(feathers, bird)	pw(beak, bird)	purpose(wing, fly)
purpose(beak, eat)	purpose(claw, catch)	sound(bird, chirp)
isa(parrot, bird)	ability(parrot, speak)	pw(straw, nest)
pw(eye, bird)	pw(leg, bird)	purpose(leg, stand)
pw(claw, leg)	role_playing(bird, freedom)	quantity(leg, 2)
isa(nest, container)	isa(house, human_setting)	

Table 2: The concept map of *bird*

or idiosyncracies. For example, we could specify that we are in face of a "new ability" if some concept X has, in the blend, the ability A, which was not present in X's input space, d1. We can even say that this "new ability" should have a minimal explanation, i.e., there must be a subpart P of X whose purpose is to provide ability A. Furthermore, we can also require that X and A be projected from different inputs ($d1$ and $d2$, resp.) to the $blend$.

$$
\begin{aligned}
&frame(new\_ability(d1)):\\
&\quad new\_ability(X,A) \longleftarrow\ ability(X,A) \wedge not\, rel(d1, ability(X,A)) \wedge\\
&\qquad\qquad purpose(P,A) \wedge pw(P,X) \wedge\\
&\qquad\qquad projection(blend, d1, X, X) \wedge\\
&\qquad\qquad projection(blend, d2, A, A)
\end{aligned}
$$

Frames can represent very abstract reasonings (e.g. the blend should have the same structure as the input space 1 - the "aframe") or very specific (e.g. the "transport_means" frame). The generic space we use in the experiments has the frames of Table 3.

In Figure 1, we give an idea of the application of frames to a blend (to improve readability, both the frames and the concepts are simplified). We say that the blend accomplishes (or satisfies) "aframe", "transport_means" and "new_ability" and that its overall *frame coverage* is 100% (every relation is included in a frame). The coverage of

Frame name	**Conditions**
aframe	The blend contains identical structure from input 1
aprojection	The blend contains the same concepts of input 1
bframe	The blend contains identical structure from input 2
bprojection	The blend contains the same concepts of input 2
pw_based_explanation	The blend contains a concept that has associated a set of part-whole relations (i.e. it is explained by a set of these relations)
transport_means	The blend contains a concept that has associated the set of features of a generic transport means
purposeful_subpart	The blend contains a a concept that has a subpart that has associated a set of relations that justify its existence (e.g. purpose, cause-effect)
new_ability	A concept has an ability relation not existent in any of the inputs
new_creature	A concept is a living thing that did not exist (or wasn't such) in any of the inputs
new_feature	A concept has a feature relation not existent in any of the inputs

Table 3: Frames of the generic space

"aframe" is aproximately 72% (the ratio of the blend that is covered by "aframe") and "transport_means" and "new_ability" have coverages of 54% and 27% (respectively).

The integrity constraints serve to specify logical impossibilities. Two examples of integrity constraints could be for specifying that something cannot be dead and alive at the same time and for avoiding part-whole recursion, i.e. something cannot have a part-whole relation (pw) with itself:

$$false \leftarrow state(X, dead) \wedge state(X, alive)$$

$$false \leftarrow pw(X, X)$$

The violation of an integrity constraint does not imply the elimination of a blend; it only brings a (configurable) penalty to its value. Thus it must have strong arguments to

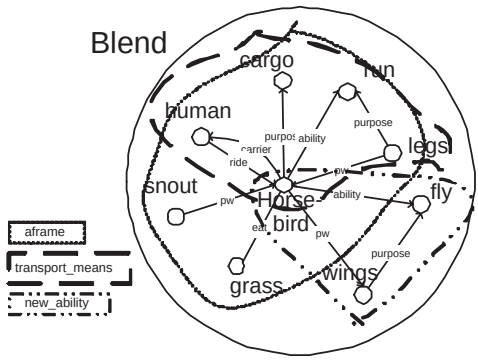

Figure 1: Three frames in a small blend

violate an integrity constraint and still be a "good blend". For space restrictions, we don't show the generic domain concept map, yet the reader should only know it has a very long list of "isa" relationships, establishing an ontological basis for the concepts (e.g. *isa(red, color), isa(human, primate), isa(physical_object, object),* etc.).

4.2 Mapper

The Mapper currently takes an optional role in the architecture. Its purpose is to generate mappings between the concept maps of the input domains automatically. It uses an algorithm of structure matching inspired in Tony Veale's Sapper framework (Veale and Keane, 1993). Basically, it uses a spreading activation algorithm to look for the largest isomorphic pair of subgraphs from the input domains. In this context, two graphs are considered isomorphic if they have the same relational (arcs) structure, independently of the concepts (nodes). There is potentially more than one structure matching between any pair of concept maps and this complexity grows worse than exponential with the number of concepts[2]. However, since it only allows alignment when it finds equal relations in both graphs, the number of possible solutions can be drastically reduced, yet still demanding Mapper to make the search in such huge space. Furthermore, the algorithm starts with a randomly selected pair of concepts, so the "perfect choice" (or even the same choice) is not guaranteed every time we run it.
This module generated three different mappings for input spaces of "horse" and "bird", as shown in Figure 2. It is important to understand that every relation has the same weight in the graph and there is no domain knowledge or special heuristics considered in the mapping construction. This means that the results may contain very unintuitive associations (e.g. "4" associated with "2"; "snout" with "bird"). The existence of "wrong" associations, however, doesn't necessarily affect the results because, when their use implies low-valued outcomes, the system will ignore them due to the "selective projection" algorithm we will describe in the next section.

4.3 Factory

Each concept x, from the input domains, has a "projection" in the blend that can be either x, nil (meaning it is not projected to the blend), y (the mapping counterpart of x) or a composition of x and y (represented by $x|y$). The latter two only possible when there is a mapping counterpart for x. To a string of "projections" of all concepts of the input domains (one "projection" for each concept), we call a "selective projection". The name "selective projection" comes from the fact that, in blends, some aspects of the input spaces are not present (i.e., not projected), some change and some remain the same.
The role of the Factory module is to make a search for blends that best fit the optimality constraints in the space of "selective projections". This space has a very high complexity. Taking a close look on this issue, we notice that, for an *input* domain 1 with m concepts and an *input* domain 2 with n concepts (with $m \geq n$), we may have the maximum of $m!$ different mappings (if we use the isomorphic mappings, as in the Mapper), with the largest mapping having a size $k = min(m, n)$. This projection selection is made independently on each concept, which means we have $l = m + n$ different concepts for each blend, each one with its own projection. So, in the "least complexity scenario", the size of the

[2] Assuming n as the number of concepts of the largest (in number of concepts) of the two concept maps, we will have a search space of $n!$ possible mappings. So, with an exponential k^n, as n approaches infinity, $\frac{k^n}{n!}$ will be 0, meaning that the search space will expand more than exponentially as the number of concepts grows

			vegetable_food	↔	vegetable
			food	↔	food
ear	↔	wing	horse	↔	bird
snout	↔	bird	equidean	↔	aves
eye	↔	lung	animal	↔	animal
mouth	↔	feathers	human_setting	↔	house
2	↔	2	wilderness	↔	wilderness
hear	↔	fly	ruminant	↔	oviparous
			run	↔	fly
	1		cargo	↔	pet
			neigh	↔	chirp
			snout	↔	lung
			mane	↔	feathers
			tail	↔	beak
mouth	↔	beak	leg	↔	eye
snout	↔	bird	paw	↔	wing
eye	↔	lung	4	↔	2
ear	↔	feathers	eye	↔	leg
eat	↔	eat	ear	↔	claw
			hear	↔	catch
	2		grass	↔	grass
				3	

Figure 2: The three mappings

mapping is 0, meaning that we have only two choices for each of the l concepts (either it gets projected to the blend or it is not projected), thus we have 2^l "selective projections". If the size of the mapping is k (the maximum possible), we have four choices for each of $2k$ concepts (k concepts in each of the domains) because each concept x mapped to y can be projected either to x, y, $x|y$ or nil. Apart from these $2k$ concepts, the rest ($l - 2k$) has only two possibilities. This leads us to the conclusion that we have a range of 2^l to $4^{2k} \times 2^{l-2k}$ different "selective projections" to choose, which is a very large search space. For example, for $m = n = 20$ (a "small" size pair of networks), we have at least 2^{40} different solutions.

Given these space dimensions and expecting that the optimality pressures (a set of competing constraints, described in next section) would complexify the search landscape, we decided to implement a parallel search algorithm, which wouldn't depend on following a specific sequence of application of those pressures. The solution we found was a genetic algorithm (GA): a framework inspired by evolutionary theory in which we have a sequence of *populations* of *individuals*, each *individual* with a *fitness* value that represents its *survival* and *reproduction* possibilities. This well-known framework has had much success in problems with a search space with the characteristics we described. The detailed formal and technical explanation of GA's is far out of the scope of this paper, so we direct the interested reader to (Goldberg, 1989). On the other side, those not interested in the technical details of our GA implementation may skip to the next section and retain the general idea that this is a parallel search that does not guarantee the "best blend" but is able to search a vast area of the space and return, with correct parameters, relevant solutions.

In our GA, the *individuals* we are *evolving* are blends, each one determined by a "selective projection". The *individual* is then an ordered sequence of projections (the *genes*), each one with an allowed value given by the projection function (from the range x, y, $x|y$ and $\emptyset$). The evaluation of a blend is made by the application of the optimality pressures, which then participate in a weighted sum, yielding the fitness value. We have populations of individuals (currently 100) that are then stochastically selected according to this value. After the selection of the individuals, the step of generation of the following population is made by using 4 operations: direct reproduction (the individual is copied to the next population); crossover (two individuals exchange part of their list of projections); mutation (random changes in the projections); random individual. The system stops when a predefined number of iterations of this process has been done, when it stabilized around a maximum for more than a predefined number of iterations or when an individual was found that has a satisfactory (predefined) value.
Through this process, Divago is able to search in a huge space of blends according to the preferences of the user. The best solution is not guaranteed, but it is reasonable to expect that the higher the number of iterations, the more likely it is to find a good blend, if one exists in the search space.

4.4 Constraints

The Constraints module implements the optimality pressures. The general role of this module is to make a preprocessing of each blend (checking frame satisfaction and completion, integrity constraint violation, vital relation projection, etc.) and then obtain a value for each of the eight measures. These values then participate on a weighted sum, which yields the *value* of the blend (normalized to fall into the [0,1] interval) that is returned to the Factory. The weight attributed to each optimality pressure is defined by the user. The optimality pressures are formalized and described in (Pereira and Cardoso, 2003), and so, an entire paper is needed to specify in detail our implementation of these. Therefore we give an informal explanation below. Beforehand, we would like to say that we make no claims in respect to the cognitive realization of each measure. These eight suggestions of quantification concern totally to the representation and scope of this model which moves towards a computational account of conceptual blending. This doesn't mean that this proposal should not be verified or tested with regard to cognition and the blending phenomena in general, it states that we didn't make our measures based on cognitive experiments, but only tried to follow the philosophy behind the description that F&T give in (Fauconnier and Turner, 2002) projected to our formal model.

4.4.1 Integration

Frames have an integration role. The reasoning behind a frame relies in the idea that concepts within it should be tightly integrated according to a situation, structure, cause-effect or any other relation that ties a set of concepts onto one, more abstract or broad, composite concept. For example, the frame of "transport_means" corresponds to a set of concepts and relations that, when connected together, fit the abstract notion of "transport means". Frames may get much more abstract and, as in the example of "aprojection" or "new_feature", represent construction directives for the blend. The integration role of these frames is not perceived as clearly as in "transport_means" but, without them, the blend will lack global consistency (as in "aprojection") or novelty (as in "new_feature"). We see frames as *information moulds* and building a blend for a given situation should depend much on the choice of these structures. Thus, the Integration value of a blend is

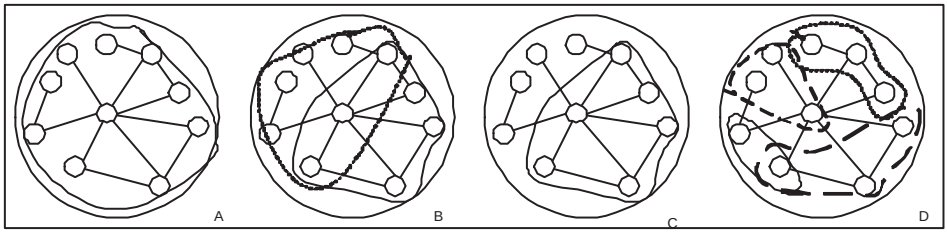

Figure 3: The role of frame coverage in Integration value

calculated with regard to a set of frames.
Assuming the set F of frames that are satisfied in a blend, we define the *frame coverage* of a domain to be the set of relations from its concept map that belong to the set of conditions of the frames in F. The larger the frame coverage of the blend, the higher its Integration value is. Yet, a blend that is covered by many frames should be less integrated than a frame with the same coverage, but with less frames. In other words, if a single frame covers all the relations of a blend, it should be valued with the maximal Integration, whereas if it has different frames being satisfied and covering different sets of relations, it should be considered less integrated. The intuition behind this is that the unity around an integrating concept (the frame) reflects the unity of the domain. In the diagrams of Figure 3, we give an informal idea of this reasoning. Blend A has the maximum Integration value (100%) because a single frame is able to cover the complete concept map. B will have a lower value since it needs two frames to cover the same area. Blends C and D will have lower values than the other two and C will have higher value than D because frame coverage of D is too much dispersed.

The Integration measure we propose also takes integrity constraints into account so that, when a frame violates such a constraint, it is subject to penalty. Integration belongs, along with Relevance, Topology and Unpacking, to the fundamental bricks of the blending process of Divago. It is intended to lead the choice of the blend to be something *recognizable* as a whole, fitting patterns that help to determine and understand what a *new* concept is.

4.4.2 Topology

The Topology optimality pressure brings *inertia* to the blending process. It is the constraint that drives against change in the concepts because, in order to maintain the same topological configuration as in the inputs, the blend should maintain exactly the same neighborhood relationships between every concept, ending up being a projected copy of the inputs. This pressure is normally one that is disrespected without big loss in the value of the blend. This is due to the *imagination* context that normally involves blends, i.e., novel associations and changes in previous ones are not necessarily penalized.
We calculate our Topology measure by simply obtaining the ratio of relations ($r(x, y)$) in the blend that also exist in the inputs. This means, for example, in blending "horse" and "bird", that obtaining an exact "horse" or an exact "bird" would give the highest Topology value (every relation would also exist in the corresponding input domain), while adding the relation "ability(horse, fly)" to the "horse" would lower that value (there is no "ability(horse, fly)" in any domain).

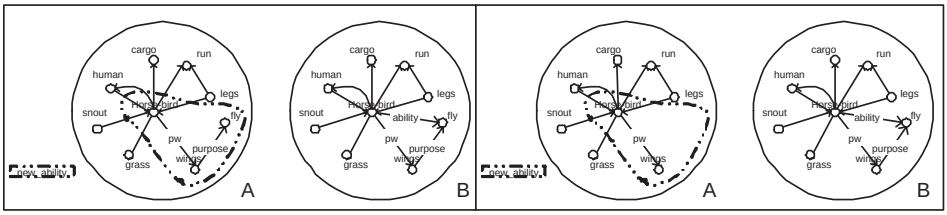

Figure 4: Pattern Completion examples

4.4.3 Pattern Completion

The Pattern Completion pressure brings the influence of patterns present either in the *inputs* or in the *generic* space. Sometimes, when reasoning about a concept (or a set of concepts), it may make sense to *complete* it with new knowledge. For example, if we have a "horsebird" defined as having "2 wings made of feathers", we may *complete* it with the "flying ability" by matching the concept of "horsebird" with the pattern of "flying creatures": "Flying creatures have 2 wings. They are made of feathers and serve to fly".
At present, in the context of this work, a pattern is described by a frame, i.e. we don't distinguish these two concepts, and therefore pattern completion is basically frame completion. Here, as in the definition of this principle, the completing knowledge becomes available from "outside", not as a result of projection. This means that the act of completing a frame consists of asserting the truth of the ungrounded premises from frames of the generic domain, a process that happens only after a sufficient number of premises is true. We call this the *evidence threshold*. In Figure 4, we show two examples of the evidence thresholds of a frame ("new_ability") with regard to two different blends. In the first one, the frame has an evidence of 67% approximately (the frame has three relations, and two are true in the blend) and so its completion is made by adding the relation "ability(horsebird, fly)". In the second one, the evidence threshold is approximately 33% (and so it is completed by adding two new relations).

As in the integration pressure, we have the problem of taking into account multiple frames. This time, given that we are evaluating possible completion of subsets of relations, instead of sets of relations that are actually verified in the domain, it is difficult to find such a linear rationale (e.g. would two patterns each with individual completion x be valued higher than three having each slightly less than x?). As a result, the value of Pattern Completion of a blend corresponds to the evidence threshold of the union of the frames (which will be the frame with all conditions that appear in all the frames).

4.4.4 Maximization of Vital Relations

For the maximization of vital relations, we estimate the impact of the inner-space and outer-space vital relations to the blend. Fauconnier and Turner list a set of 15 vital relations (in footnote of page 3 of this document) that exist between elements within a mental space (inner-space relations) and between different mental spaces (outer-spaces). These are *special* relations in the sense that they can be *compressed* in blend. Since in our current model we don't explicitly approach compression, vital relations will be a set of (customizable) relations whose projection should be valued.
The measure of Maximization of Vital Relations is calculated as a ratio of the actual num-

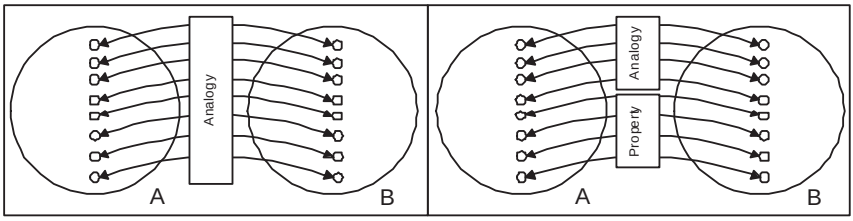

Figure 5: Intensification VR examples

ber of vital relations in the blend w.r.t. the maximum possible number of vital relations (that would appear in the blend if every concepts were projected).

4.4.5 Intensification of Vital Relations

The difference between Intensification and Maximization of Vital Relations is not clear in the definition so we propose a perspective that may disagree with what was originally meant by Fauconnier and Turner. In our case, Intensification of Vital Relations is a measure concerning (exclusively) outer-space relations, more specifically the relations established by the mapping algorithm. The rationale is that, for each vital relation, there is a mapping algorithm that connects elements from the two input spaces with the respective vital relation and the use of these connections will yield an estimate of how "intense" this vital relation is in the blend. In our case, the (only) vital relation established is "analogy" (e.g. there is an "analogy" relation between "run" and "fly" in mapping 3, Figure 2) and its intensity is measured by the *systematicity principle* (if x is associated to y, its neighbors should also be associated). For different vital relations (e.g. "disanalogy"), different intensity measures could be applied. We have only implemented "analogy" so far, so our proposal for this measure is far less solid than the others.
The calculation of the value for Intensification pressure takes the point of view that a blend that applies mappings generated by only one vital relation (suppose it has an intensity value x) should have higher measure than a blend that apply n vital relations (suppose each with intensity value x/n). We want to favor "concentration", therefore there is a penalty for the proliferation of different vital relations. In Figure 5, we show an example of two choices for mappings. The mapping on the left will get higher Intensification value because it is concentrated around a single vital relation ("analogy").
In the experiments, our mapping was based on a single vital relation therefore this measure could not yet be tested.

4.4.6 Unpacking

Unpacking is the ability to reconstruct the whole process starting from the blend. From our point of view, such achievement underlies the ability to reconstruct the input spaces. The reconstruction of the input spaces from the blend demands the assessment of the cross-space mappings, the generic space and other connections. Thus, what we are proposing is that Unpacking can be reduced to the ability to reconstruct the inputs. This is so because there is no way to properly reconstruct the inputs without a reconstruction of the cross-space mappings, generic space and the connections between spaces.
Unpacking should take the point of view of the "blend reader", i.e., someone or something that is not aware of the process of generation, thus not having access to the actual projections. Being such, this "reader" will look for patterns that point to the "original"

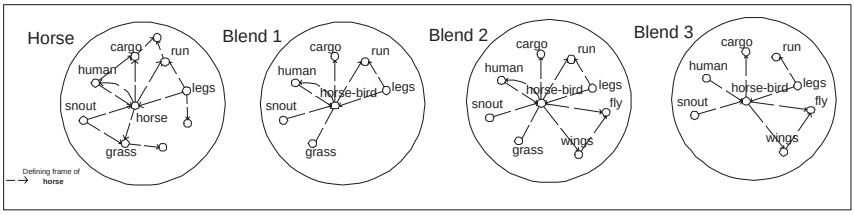

Figure 6: Unpacking examples

concepts. Once again we use the idea of *frames*, more specifically the *defining frame* of a concept, which comprises its immediately surrounding relations. In the blend, if we can identify clearly the defining frames of the original concepts, then its Unpacking value is high. In Figure 6, we present the defining frame for "horse", in the "Horse" domain. In Blend 1, the concept "horse-bird" (the projection of "horse") will have the highest Unpacking value because it fits exactly its defining frame. In Blend 2, the value is lower because there are two new relations (with "fly" and "wings"), meaning it is not the exact same concept. Blend 3 will get the lowest Unpacking value of all three because it also lacks some relations (e.g. with "run" and "grass").

The calculation of the Unpacking value of a blend corresponds to an average of the individual Unpacking values of all the concepts in the concept map. Each of these individual Unpacking values takes into account the completion ratio of the defining frame and the extra relations added.

4.4.7 Web

The Web principle concerns to being able to "run" the blend without cutting the connections to the inputs. It is our opinion that this is not an independent principle, being co-related to those of Topology and Unpacking because the former brings a straightforward way to "maintain the web of appropriate connections to the input spaces easily and without additional surveillance or computation" and the latter measures exactly the work needed to reconstruct the inputs from the blend. It is not to say that Web is the same as Topology or Unpacking, what we are arguing is that, on one side, Topology provides a pressure to maintain the most fundamental connection to the input: the same set of relations; on the other side, Unpacking evaluates the easiness of finding the connections to the inputs. The weighted sum of these two values yield, we propose, an estimation of the strength of the web of connections to the inputs.
Not being an independent variable, we don't apply the Web constraint in the tests we show here.

4.4.8 Relevance

The notion of "relevance" or "good reason" for a blend is tied to the pragmatics of the situation, or, in other words, the context and goal of the blending generation. Once again, frames take a fundamental role, they are "context specifiers" (i.e., the set of constraints within a frame describe the context within which the frame is fulfilled). Thus, Divago allows the specification of a *query* that may contain a set of conditions (e.g. if we want to find a concept that "flies", we could add the condition $ability(x, fly)$) and a set of frames to be accomplished (e.g. the blend should accomplish the "transport_means" frame). This query will then correspond to a set of frames (the set of conditions is also considered a

frame itself) to which we call *goal frames*.
Having a set of goal frames, which could be selected from any of the existent domains or specified externally, a blend gets the maximum Relevance value if it is able to satisfy all of them. In this measure we must also take into account partial completion of the goal frames. A blend that "almost" satisfies a goal frame should be valued in relation to a frame that doesn't (assuming both are equal in the other features). Regarding this, we consider a factor for the partial completion of the goal frames following the same procedure as in Pattern Completion.
Intuitively, this measure takes two parts: the satisfied goal frames and the unsatisfied goal frames. The value of the latter depend on completion (e.g. if Completion=50%, these count as "half" satisfied goal frames).
The Relevance principle allows a user to specify his notion of *usefulness*. In fact, the usefulness of a concept is always a pragmatic matter. Something can be extremely and obviously useful in a context and the opposite in another. For example, in a graphical environment, useful concepts should have a color, a shape, a position and so on, while in a text-based role-playing game, they are expected to have the game's attributes (e.g. mood, strength, role, etc.). Thus, a blend that has a 100% Relevance value satisfies all the conditions of the query, meaning it contains all the knowledge needed for a given context, i.e., it is useful with regard to that context. This is not to say that usefulness is easily specified, as it is not straightforward to find all constraints and requirements that need to be respected in a given context.

5 Experiments

We made two different experiments, each with a distinct goal: assessment of the individual effects of each measure on the final results; qualitative evaluation and tuning of the model. After several preliminary GA parameters tuning tests, we decided for 100 individuals as the population size, 5% of asexual reproduction (copy of an individual to the following population), 80% of crossover (combination of pairs of individuals), 5% of mutation and 1% of random generation (to allow random jumps in the search space). We have three different stopping conditions: appearance of an individual with the maximum value (1); achieving n populations ($n = 500$); being stalled (no improvements in best value) for more than m populations ($m = 20$). We kept these GA configurations throughout the two experiments.

5.1 Evaluating Optimality Pressures

This test serves to observe the real effect of each pressure in the final results, bringing up a way to predict and control the system. For the first part of these experiments, we isolated each optimality pressure, by attributing zero weight to the remaining criteria. Since one of the optimality pressures is not independent (Web) and another (Intensification of V.R.) only applies one mapping algorithm (based on analogy), we did not test them, so we had six different criteria to take into account.
The input domains we applied were the domains of *horse* and *bird* (in Tables 1 and 2), meaning that the expected results range from the unchanged copy of one (or both) of the concepts to a horse-bird (or bird-horse) which is a combination of selected features from the input domains. The generic domain consists of a simple general ontology (essentially an "isa" tree with concepts from high-level like "information_entity" or "physical_object" to low-level like "digit" or "dog"), a set of frames and integrity constraints (see Table

3). We applied the mappings presented in Figure 2. For each mapping, we tested the six optimality pressures, each of these comprising 30 runs[3].
We present now a detailed analysis of the individual effect of each of the measures:

- In **Integration**, frames behave as *attractor* points in the search space. Moreover, the frames with a larger coverage tend to be preferred, although when too large (like *aprojection* or *aframe*) they are dropped away. The evolution is directed to a compromise of coverage and satisfiability. The complexity of the search space grows with mapping size (the number of cross-space associations found by the mapping algorithm). In fact, when we have a mapping of size 5, it returns six different blends, being the best choice retrieved 43% of the times, while with a mapping size of 21, it finds eight different solutions, being the best choice retrieved only 6% of the times. This confirms the complexity and dimensions of the search space we discussed in section 4.3. A good compensation for this apparent loss of control is that the returned values are clearly higher (0.68, for the best) than in the small mappings (0.22), suggesting that, with larger mappings, the probability of finding a better solution is higher than in smaller ones.

- **Pattern Completion** drives the blend to partially complete (i.e., satisfy some of conditions but not all) the highest possible number of frames, leading, in each case, to several sets of relations that fit into those frames without satisfying them. This means that, isolated, Pattern Completion only leads to disperse, non-integrated results and so it is not very useful. Interestingly, it can be useful when combined with Integration because it brings gradually to the blend the concepts and relations that are needed to complete the frames and so speeding up the process of finding frames with high Integration value. In which respects to the *search landscape*, it seems to be very rich in local maxima. The most constant results came from mapping 2 (of Figure 2), with the best results obtained in 13% of the times and the second best in 20%. An interesting remark is that the resulting local maxima always fall within a very strict range of values (of maximum amplitude 0.11, in mapping 3).

- In all the experiments with **Topology**, the final results were valued 100%, meaning that this constraint is easily fully accomplished, independently of the mapping. An interesting fact is that there is a multitude of solutions in the *search landscape* of Topology, showed by the amount of different final results in each mapping. Intuitively, and observing the short duration of each run, this means that, wherever the search starts, there is always a Topology optimal point in the neighborhood. From observation of the relations contained in the final results, we see that this constraint brings a tendency for *disintegration*, i.e, small isolated graphs appear in the blend. Each isolated graph is either a copy of a (normally unmapped) subgraph of one input source or consists of complete structure matching (there are concepts from both domains, but only the relations that exist in both are present)

- The influence of **Maximization of Vital Relations** in the results is straightforward, given that its highest value (1) reflects the presence, in the blend, of all the vital relations that exist in the inputs. As the evolution goes on in each run, the value grows until reaching the maximum reasonably early. For each set of 30 runs, it reached the value 1 a minimum of 93% of the times, and the remaining 7% achieved

[3] A run is an entire evolutive cycle, from the initial population to the population in which the algorithm stopped

at least a value of 0.95. As in Topology, the search space of Maximization of Vital Relations is very *simple* since there is a global maximum in the neighborhood of (almost) every point.

- The results of the **Unpacking** measure show that it has a deleterious side effect on the size of the blend, it drives it to very small sets (between 0 and 5) of relations. The interpretation here is straightforward: the ratio of *unpackable* concepts is highly penalized in bigger sets because of the projected relations that come as side effect of the projection of (*unpackable* or not) concepts. These relations *confuse* the unpacking algorithm so that it leads the evolution to gradually select the smaller results. The maxima points also correspond to the value 1, but it seems, from the experiments, that there is a very limited set of such individuals, achieved in the majority (at least 93% for each mapping) of the experiments.

- The first part of the test on **Relevance** focussed on making a single relation query. In this case, we asked for "something that flies" (ability(_, fly)). The results were straightforward in any mapping, accomplishing the maximum value (1) in 100% of the runs, although the resulting concept maps did not reveal necessarily any overall constant structure or unity, giving an idea of randomness in the choice of relations other than ability(_, fly). In other words, the evolution took only two steps: when no individual has a relation "ability(_, fly)", therefore with value 0; when a relation "ability(_,fly)" is found, yielding a value 1, independently of the rest of the concept map. The second part of the test on Relevance, by adding a frame (*ability_explanation*) to the query, revealed similar conclusions. There was no sufficient knowledge in any of the input domains to satisfy this new frame completely, so the algorithm searched for the maximum satisfaction and reached it 100% of times in every mapping. So the *landscape* seems to have one single global and no local maxima, reflecting the integration of the two parts of the query. If there were separate frames, it is expectable the existence of local maxima. Intuitively, the *search landscapes* of Integration and Relevance seem to be similar.

It is important to stress that, in the current version of Divago, no inference is done within the blends. Each blend is examined by the Constraints module without being subject to any transformation after the projections. In other words, there is yet no "running the blend", an aspect that will be focussed in the next developments of this work.

5.2 Qualitative evaluation

In this stage of the experiments, we tried to understand the behavior of the system by generating and observing different blends, each one with a specific goal. The first goal was to generate a *well known* blend of a horse and a bird: the *pegasus*. Then, we allowed more variations of this creature, by changing the mapping or the weights of the optimality pressures. Finally, we tried to generate different creatures that, from our point of view, reveal interest.

5.2.1 The Pegasus

For our concerns, we define a pegasus as being a "flying horse with wings", so leaving out other features it may have (such as being white). These extra features could also be considered but would need knowledge concerning to the several aspects of ancient Greece, Greek mythology and some ontological associations (e.g. purity is white). Moreover,

they would make the generation of the blend considerably more complex, even if more interesting. Formally, the pegasus we want to generate has the same concept map as the horse domain augmented with 2 wings and the ability to fly (the relations "ability(horse, fly), motion_process(horse, fly), pw(wing, horse) and quantity(wing, 2)").

For validation purposes, we started by submitting a query with all the relations of the pegasus, so as to check if they could be found in the search space, and obviously the results reveal that only the mapping 3 (see Figure 2) respects such constraints. This led us to use exclusively this mapping throughout this section.

Knowing that the solution exists in the search space, our goal was to find the minimal necessary requirements (the weights, the frames and the query) in order to retrieve it. From a first set of runs, in which the system considers a big set of different frames and no query, we quickly understood that it is not simple (or even possible) to build the pegasus solely by handling the weights. This happens because there is no controlling device that allows a user or an evaluation function to drive the evolution to a particular place. The optimality pressures provide control regarding to structural evaluation and general consistency and may yield interesting results, but only by chance a pegasus, which drives us to the need of queries.

A query may range from specific conditions that we demand the blend to respect (e.g. the set of conditions for flying, enumerated above) to highly abstract frames that reflect our preferences in the blend construction (e.g. the frame *aprojection*: elements from input space 1 should all be projected). Intuitively, the best options seem to comprise a combination of the different levels of abstraction.

Since a query is only considered in the Relevance measure, its weight must be large if we intend to give it priority. In fact, using only Relevance is sufficient to find the solution if the query is specific enough, as we could test by using a query with *aprojection* and the flying conditions. From a creativity point of view, it is not expected to have very specific queries (in these cases, the search wouldn't be needed, in the first place) and we are more interested in less constrained search directives. In the Table 6, we show the parameters we used. The weights we present correspond to Integrity (I), Pattern Completion (PC), Topology (T), Maximization of Vital Relations (MVR), Unpacking (U) and Relevance (R). The "fly conds." are the relations the blend must have in order to be a flying creature, and aframe, aprojection and new_ability are frames as described before.

Exp.	Weights						Query
#	I	PC	T	MVR	U	R	
1	0	0	0	0	0	1	fly conds. + aprojection
2	0	0	0	0	0	1	fly conds. + aframe
3	0	0	0	0	0	1	fly conds.+ aprojection + aframe
4	1	0	0	0	0	1	fly conds.+ aprojection + aframe
5	1	1	0	0	0	1	fly conds.+ aprojection + aframe
6	1	0	1	0	0	1	fly conds.+ aprojection + aframe
7	1	0	1	1	0	1	fly conds.+ aprojection + aframe
8	1	0	1	1	1	1	fly conds.+ aprojection + aframe
9	8.5	0	4	2.5	1	9	fly conds.+ aprojection + aframe
10	8.5	0	4	2.5	1	9	new_ability+aframe+aprojection

Table 4: Parameters used in each experiment.

The first eight experiments were dedicated to understanding the effect of gradually adding optimality pressures to the fitness function. In the first three, where only Relevance

was used, we verified that, although it was *easy* to have all the concepts and relations we expect for a pegasus, often it was complemented by an apparently random selection of other relations. This results from having no weight on Integration, which we added on the experiment 4, yielding the most strict pegasus, the projection of the entire horse domain, and the selective projection of wings and the fly ability from the bird domain, in more than 90% of the runs. In experiment 5, the influence of Pattern Completion led the results to minimum incompleteness (e.g. a pegasus with everything except a mane, wings or any other item), which revealed that, by itself, it is not a significant or even positive contribution to the present goal, a reason for dropping its participation in the following experiments. Moreover, it suggests a revision of the implementation of this measure.

Adding Topology (exp. 6) brought essentially two different kinds of results. In 60% of the runs, it returned the "correct" pegasus with extra features like having feathers or a beak (which was not constrained in the query), either of each apparently selected at random. These were also given the higher scores in the experiment. In other 37% of the runs, the results were either "simple" horses or a compromise between a bird and a horse (e.g. two legs, a beak, two wings, ruminant, a mane, paws, etc.). A possible interpretation is that, on one side, the frames *aprojection* and *aframe* already imply strong topological maintenance, and Topology itself brings knowledge that, although not considered in the frames, strengthens this value. Yet, this does not avoid the existence of local maxima that represent stable results, in terms of the weights considered. The following experiment, the inclusion of Maximization of Vital Relations, confirmed the same conclusions, but with more control over the kind of extra relations transferred to the blend. For example, the blend may have 2 wings (from the relation *quantity*), a beak and feathers (from *pw*), but it is never an oviparous (from *taxonomicq*). On the other hand, we can sense a gradual lack of focus on the overall results (no two runs returned the exact same result) complicating considerably our goal of controlling the system. There is a simple explanation for this: Relevance, Integration, Topology and Maximization of V.R. all have the same weight and some (like Maximization) are more easily satisfied, thus driving the evolution around their maxima.

The eighth experiment brought a more stable set of results. Adding Unpacking to the other pressures reassures the prominence of the "basic" pegasus, but, as happened with the majority of sixth experiment results, augmented with features projected from the bird domain. This time, some of these new features came isolated to the blend, i.e., not connected to the rest of the blend (e.g. there are 2 claws that serve to catch, but they don't make part of anything).

An immediate conclusion we took from these first 8 experiments was that each pressure should have a different weight, correspondent to the degree of influence it should have in the result. In our case, we are seeking for a specific object (the pegasus), we know what it is like, what it should not have and some features not covered by the query conditions that we would like it to have. This led us to a series of tests for obtaining a satisfiable set of weights, used in the configurations 9 to 12. Given the huge dimension of the problem of finding these weights, they were obtained from a generate-and-test process, driven by our intuition, so there is no detailed explanation for the exact choice of these values and not others. Yet, a qualitative analysis can be made and we see a clear strength given to Relevance and Integration. The former serves to "satisfy what we asked" and the latter guarantees overall coherence (centered on the query frames) and consistency (e.g. it prevents the solution from having 2 and 4 legs simultaneously). There is also a more discreet presence of Topology, Maximization and Unpacking, to allow the transfer of extra knowledge.

The experiment 9 revealed, possibly, the "best" pegasus we could expect. As we can see

quantity(wing, 2)	conditional(wing, fly)	motion_process(horse, fly)
ability(horse, fly)	purpose(wing, fly)	pw(wing, horse)
isa(horse,equinae)	pw(leg, horse)	purpose(horse, food)
isa(equinae,mammal)	purpose(leg, stand)	sound(horse, neigh)
existence(horse, farm)	pw(paw, leg)	purpose(mouth, eat)
existence(horse, wilderness)	purpose(horse, traction)	purpose(ear, hear)
pw(snout, horse)	eat(horse, grass)	color(mane, dark)
pw(mane, horse)	ability(horse, run)	size(mane, long)
pw(tail, horse)	carrier(horse, human)	material(mane, hair)
quantity(paw, 4)	quantity(leg, 4)	purpose(horse, cargo)
pw(eye, snout)	quantity(eye, 2)	taxonomicq(horse, ruminant)
pw(ear, snout)	quantity(ear, 2)	ride(human, horse)
pw(mouth,snout)	purpose(eye, see)	motion_process(horse,walk)

Table 5: Example 1 (from experiment 9)

purpose(claw, catch)	pw(claw, leg)	purpose(lung, breathe)
pw(lung, horse)	conditional(wing, fly)	motion_process(horse, fly)
ability(horse, fly)	purpose(wing, fly)	pw(wing, horse)
isa(horse,equinae)	pw(leg, horse)	purpose(horse, food)
isa(equinae,mammal)	purpose(leg, stand)	sound(horse, neigh)
existence(horse, farm)	pw(paw, leg)	purpose(mouth, eat)
existence(horse, wilderness)	purpose(horse, traction)	purpose(ear, hear)
pw(snout, horse)	eat(horse, grass)	color(mane, dark)
pw(mane, horse)	ability(horse, run)	size(mane, long)
pw(tail, horse)	carrier(horse, human)	material(mane, hair)
quantity(paw, 4)	quantity(leg, 4)	purpose(horse, cargo)
pw(eye, snout)	quantity(eye, 2)	taxonomicq(horse, ruminant)
pw(ear, snout)	quantity(ear, 2)	ride(human, horse)
pw(mouth,snout)	purpose(eye, see)	motion_process(horse,walk)

Table 6: Example 2 (from experiment 9)

in the two results presented in tables 7 and 8, it has all the horse features, the specified "flying" requirements and some added knowledge that we consider valid, like having 2 wings, lungs or claws. It is clear that these results were subjectively driven by us in the choice of the concepts and frame design, but the argument we try to bring is that it produces a new concept that, not only respects the query, but also brings new knowledge that was selectively projected.

In the final experiment (10), we decided to give a more vague specification, asking only for a *new_ability* in the blend, as well as the generic frames *aprojection* and *aframe*. As a result, we found the exact pegasus in 23% of the times. This gives the first evidence that the system can be used for generating concepts without a very constraining and specific query and led us to the following experiments, in which we tried to assess its generative possibilities.

5.2.2 Other creatures

In order to explore the potential of the system, we made additional tests, without imposing specific goals beforehand. We didn't make significant variations on the weights of the

previous tests. For two, we removed some weights from the configuration and reduced Integration in the latest ones. In table 9, we show all the configurations (the omitted weights are 0, as in the other experiments). We made variations on the query and checked the results, trying not to bias for particular outcomes. Therefore, these tests aim to give an informal insight on the generative potential of the system.

We found several "creatures" that we'd like to describe. To the first (experiment 11), we call "dumborse", a flying horse that uses its ears as wings (like *Dumbo*, the flying elephant). This "creature" is possible to find in mapping 1 (*ears* are mapped onto *wings*). It is exactly a horse, but it has wings instead of ears, which serve to fly and to hear. With *Dumbo* in mind, we tried to go further to a horse with ears that serve to fly and hear (instead of wings in place of ears), and this was achieved by allowing only weights on Integration and Relevance (experiment 12). A simple explanation is that, while it satisfies entirely Relevance and, almost totally, Integration, it has less topology and less Unpacking (ears don't ever relate to fly in the bird domain).

Another creature to report is the "flying snout" (which appeared in 23% of the runs of configuration 13, see table 9), a snout that has all the features of the bird. This is a "weak" blend in the sense that an isolated concept (the "horse snout") gets projected to the "bird" structure without any surrounding support such as its shape or its purpose. The third creature is the transport bird, which has all the features of the bird, but also carries humans, it serves for cargo and traction. It appeared occasionally during the previous experiments, but was triggered now by the frame "transport_means" in the query (in configuration 20), meaning indeed we had it in mind. Yet, its appearance throughout the tests (only when dealing with mapping 3, though) led us to include it in this section. The fourth creature is an oviparous horse, with two legs (instead of four), two wings and claws. It appeared in less than 10% of the results in the configuration 20, but it was the one that got the highest score.

In configurations 14, 15, 21 and 22, the results were essentially copies of the "bird" concept map, whereas 19 and 21 yielded highly unstable partial projections of both the "horse" and the "bird" concept maps simultaneously to the blend, since each of the 30 runs returned a different concept map. In the latter, we find it difficult to interpret anything. A possible explanation for these unsuccessful configurations is that the frames used are too much abstract, leaving no concrete goal to the system.

Exp.	Weights					Query	Mapping
#	I	T	MVR	U	R		
11	8.5	4	2.5	1	9	new_ability+aframe+aprojection	1
12	8.5	0	0	0	9	new_ability+aframe+aprojection	1
13	8.5	0	0	0	9	new_ability+ bprojection + bframe	1
14	8.5	4	2.5	1	9	new_ability + bprojection + bframe	1
15	8.5	4	2.5	1	9	bprojection + bframe	1
16	8.5	4	2.5	1	9	new_ability + bprojection + bframe	3
17	8.5	4	2.5	1	9	bprojection+ aframe	1
18	8.5	4	2.5	1	9	bprojection+ aframe	3
19	8.5	4	2.5	1	9	aprojection+ bframe	3
20	4	4	2.5	1	10	transport_means+bframe+bprojection	3
21	4	4	2.5	1	10	transport_means+bframe+bprojection	1
22	4	4	2.5	1	10	transport_means+bframe+bprojection	5

Table 7: Parameters for configurations 11 to 22

These ad-hoc experiments reveal that the system can produce novel concepts, yet it also demonstrates clearly that we face a very large search space, demanding a serious reflection about the tuning of the system.
It is the capacity to create novel and valid (with regard to the queries) creatures that testifies the potential of this model towards computational creativity. On one hand, it surely allows the creation of new concepts, a vital feature of a creative process, but on the other hand, the ultimate control always needs to be parameterized by a user (or another system?). There seems to be a paradox here: one must orient the system towards novelty and usefulness, but if doing so exhaustively, the emergent *creativity* is set *a priori* by the parameters. Yet, this apparent paradox seems to be present in discussions around creativity regarding issues like intentionality or evaluation. In fact, the boundaries between what is and what is not a creative product are very controversial and fragile. In our case, this boundary may lie within the level of abstractness given in the specification, which should comprise the mandatory conditions (e.g. specific frames) and more abstract preferences (e.g. abstract frames, like *aframe*).

6 Discussion

As we expected, the experiments raised several fundamental issues, some of which demanding a short reflection. Does the system agree totally with the Conceptual Blending framework? Does this system implement any kind of Computational Creativity? What can we expect from this model?
Since Fauconnier and Turner do not present a formal perspective on Conceptual Blending, it is not straightforward to validate our work in this respect. Starting from the representation of a mental space, we decided for a static, generic notion, the *domain*. We believe the representation we use (or an extension of it) could lead to mental spaces in general, but we are not confident to claim so much yet. This reduction of the knowledge basis of Conceptual Blending - the mental space - brings, *a priori*, limitations to our model. If successful, it should be able to produce the specific types of blends that result from blending static knowledge, such as domains, as opposed to dynamic knowledge, such as we have in discourse. In the latter case, we would need to extend our language to consider modalities, tense, mood, perspectives, or any other subjective, pragmatic or circumstantial components of discourse. Assuming that concepts, like “horse” and “bird”, can be validly defined by domains (those of “horses” and “birds”, from a common sense perspective), our model is expected to generate new concepts, like “horse-bird”, described in the same language. Above all, this “horse-bird” must be understandable from *a)* the chain of explanatory connections that appear in the new domain; *b)* the reference to the input domains, in the end-points of the explanatory connections. This agrees with the notions of *emergent structure* and *web of connections* that are present in the fundamentals of Conceptual Blending.
In (Pereira and Cardoso, 2002b) and (Pereira and Cardoso, 2002a), we discussed the potential of this framework from the point of view of Computational Creativity, namely in transforming the search space by changing the meta-level description of a domain. We showed that, having a level of instances (in the example of that paper, visual objects of a “house” and a “boat”), a theory for explaining the concepts involved in them, and assuming these instances as the search space for the problem (in the example, “drawing a house” or a “boat”), it is possible to obtain new ideas via blending the theories. After generating a blend (with the first version of this model, formally described in (Pereira and Cardoso, 2001)), the system reinterpreted each of the instances according to the new

relations (e.g. a house with a round window). There were no criteria for assessing the value of the blends or even selective projection. The idea was to generate the whole new search space at the instance level starting from different mappings. Currently, we focus on domain theories and on the evaluation of the blends via optimality pressures, leading to further conclusions about the creative aspects of this model. Creativity has, without controversy, two important aspects: novelty and usefulness. The work described in (Pereira and Cardoso, 2002b) and (Pereira and Cardoso, 2002a) is centered on novelty, leaving the task of choosing the "useful" results a responsibility of the search procedure. The combination of the two could then be novel and useful. A step further, the model we present now brings two components that may be valuable for usefulness: the frames and the optimality pressures. Frames provide low-level specifications or directives that should be valued in the blend, whereas the 8 optimality pressures work as high-level directives that allow the system to evaluate each blend according to several aspects. Thus, without having to exhaustively specify the query, it is possible to generate a novel concept that conforms a set of constraints. From the assumption that the ability to create concepts is a factor of creativity, we argue that ours is a computational model of creativity.

7 Acknowledgements

The authors would like to thank the reviewers for the patience and effort they spent for giving their very constructive commentaries for this paper.

References

Andersen, C. F. (1996). *A Computational Model of Complex Concept Composition*. University of Texas at Austin.

Costello, F. J. and Keane, M. T. (2000). Efficient creativity: Constraint-guided conceptual combination. *Cognitive Science*, 24(2):299–349.

de Saussure, F. (1983). *Course in General Linguistics*. London:Duckworth.

Fauconnier, G. and Turner, M. (1998). Conceptual integration networks. *Cognitive Science*, 22(2):133–187.

Fauconnier, G. and Turner, M. (2002). *The Way We Think*. Basic Books.

Freeman, M. (1999). The role of blending in an empirical study of literary analysis. In *Proceedings of the 6th International Cognitive Linguistics Conference*.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley.

Hutchins, E. (2002). Material anchors for conceptual blends. *forthcoming*.

J. Bateman, B. M. and Fabris, G. (1995). The generalized upper model knowledge base: Organization and use. *Towards very large knowledge bases: knowledge building and knowledge sharing*.

Lakoff, G. and Nunez, R. (2000). *Where Mathematics Comes From: How the Embodied Mind brings Mathematics into Being*. New York:Basic Books.

Lenat, D. B. (1995). Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11).

Miller, G. (1995). Wordnet: A lexical database for english. *Communications of the ACM*, 38(11).

Murphy, G. L. and Medin, D. L. (1985). The role of theories in conceptual coherence. *Psychological review*, (92).

Ogden, C. K. and Richards, I. A. (1923). *The Meaning of Meaning*. London: Routledge and Kegan Paul, Ltd.

Pereira, F. C. (1998). Modelling divergent production: a multi domain approach. In *European Conference of Artificial Intelligence, ECAI98*. IOSPress.

Pereira, F. C. (2003). Experiments with free concept generation in divago. In A. Cardoso, C. B. and Gero, J., editors, *Proceedings of the Third workshop on Creative Systems*. IJCAI. Forthcoming.

Pereira, F. C. and Cardoso, A. (2001). Knowledge integration with conceptual blending. In *Proceedings of AICS 2001*.

Pereira, F. C. and Cardoso, A. (2002a). The boat-house visual blending experience. In *Proceedings of the Symposium for Creativity in Arts and Science of AISB 2002*. AISB.

Pereira, F. C. and Cardoso, A. (2002b). Conceptual blending and the quest for the holy creative process. In *Proceedings of the Symposium for Creativity in Arts and Science of AISB 2002*. AISB.

Pereira, F. C. and Cardoso, A. (2003). Optimality principles for conceptual blending: A first computational approach. In *AISB'03 Symposium on AI and Creativity in Arts and Science*. SSAISB.

Veale, T. (1997). Creativity as pastiche: A computational treatment of metaphoric blends, with special reference to cinematic "borrowing". In *Proceedings of the Mind II: Computational Models of Creative Cognition*. Dublin, DCU.

Veale, T. and Keane, M. (1993). A connectionist model of semantic memory for metaphor interpretation. In *Neural Architectures and Distributed AI Workshop*. Centre for Neural Engineering, U.S.C. California.

Veale, T. and O'Donogue, D. (2000). Computation and blending. *Cognitive Linguistics*, Special Issue on Conceptual Blending.

AISB Journal **1**(3), © SSAISB, 2003
http://www.aisb.org.uk

Quantitative Abstraction Theory

Chris Thornton

Cognitive and Computing Sciences, University of Sussex
Falmer, Brighton, BN1 9QH, UK
chris.thornton@firenet.uk.com

Abstract

A quantitative theory of abstraction is presented. The central feature of this is a growth formula defining the number of abstractions which may be formed by an individual agent in a given context. Implications of the theory for artificial intelligence and cognitive psychology are explored. Its possible applications to the issue of implicit v. explicit learning are also discussed.

1 Introduction

Abstraction has long been assumed to be a key process in cognition. And though it has never been given a generic specification, philosophers since the time of Aristotle have been willing to accord it a central role. In Aristotle's case (1), and later in Locke's (2), the process was seen as lying at the heart of the problem of 'universals'.[1] More recently, artificial intelligence researchers have carried on the tradition, letting abstraction take the strain in models of search, problem solving, theorem proving, planning, reasoning and programming [c.f. 3, Gunchiglia and Walsh, 1990, 4, 5].

That such an important process has never been put on a formal, theoretical footing is a little odd. It may be that it is regarded as too obvious and straightforward to require formuleaic treatment. And, certainly, there has been little dispute down the ages about the nature of the process itself. Accounts of the process have tended to show strong commonalities. For example, consider Hume's description from 'The Essay' (6).

> 'Tis evident that in forming most of our general ideas, if not all of them, we abstract from every particular degree of quantity and quality, and that an object ceases not to be of any particular species on account of every small alteration in its extension, duration and other properties. (pp.16-7).

Hume saw abstraction, then, in terms of the filtering-away of information of specifics, with the aim of extracting content or meaning. Aristotle also saw it this way. So too did Locke, the philosopher perhaps most strongly associated with the idea that universals are derived by abstraction from empirical data. Locke, in fact, deemed abstraction to be the leaving out of particular circumstances of time and place.

But even a brief examination of these portrayals of the process reveal underlying inconsistencies, contraditions and ambiguitities. Philosophers may agree that abstraction involves the elimination of relatively specific information. But they are less clear how

[1] One characterisation of the difference between Plato's and Aristotle's views on the derivation of universal truths was that Plato saw them as coming 'from above' while Aristotle saw them as coming 'from below'.

the specificity is to be measured or how the information is to be represented. In fact, any attempt to specify what is to be eliminated tends to fall foul of counter-examples. Locke's notion, for instance, that all factors relating to time and place should be eliminated would seem quite inappropriate in the case of abstraction applied to higher-level concepts, such as might relate to physical beauty for example.

Suffice it to say, then, that philosophical accounts of the process of abstraction are characteristically *pre-theoretical*. They assume the existence of a well-defined, shared meaning for the term. Since this does not exist, these accounts lack precision and are insufficient for the mechanistic and programmatic purposes of AI.

No surprise, then, that AI projects which attempt to harness the power of abstraction in a particular problem domain typically start by providing a mechanistic definition of the process (cf. Gunchiglia and Walsh, 1990). Clearly, AI researchers are aware of the fact that abstraction has no *generic* specification and that they cannot hope to make use of it without first providing a working definition.

The formulation of a generic specification for the process is a worthy goal, then, which might yield benefits right across the landscape connecting cognitive science to epistemology. It could provide a generic basis for the diversity of abstraction-using AI techniques. It might also provide a means of integrating abstraction-related ideas arising in different areas. Possibly, it might also help to fertilise new techniques for exploiting abstraction within cognition. Last but not least, it would help to further the theoretical development of artificial intelligence.

But while the present paper takes this ambitious goal as its general context, it makes no claim to reach the target or even to approach it very closely. Rather, it addresses the special problem of *abstraction quantification*. The paper shows, in particular, how we may calculate the number of abstractions which may be generated by an individual agent in a particular context. In so doing, it develops and uses a partial formalisation of the process itself. This turns out to have a number of practical and explanatory applications within AI and in related areas such as cognitive psychology. There is also the hint of a new angle on the longstanding problem of universals.

2 Derivation of the theory

Informal characterisations of abstraction (such as Hume's) normally focus on the reductive aspects of the process, i.e., the way in which relatively specific information is eliminated. But the process may also be characterised in terms of its *constructive* function. An abstraction is necessarily an abstraction *of* something. In essence, then, it is an identification of a phenomenon — an object, process or property of the abstracting agent's world. At the point of construction, the constituents must be already available. We may view abstraction therefore not in terms of the elimination of irrelevant components, but in terms of the selection and combination of relevant constituents.

The advantage of the constructive interpretation is that it opens up the possibility for quantitative analysis. Since the result of any act of abstraction is the identification of a new phenomenon embodying some combination of currently identified phenomena, we can use combinatorial reasoning to determine the number of abstractions a given agent can form starting from a base of primitive identifications.

However, there are several complications to take into account. The number of possible abstractions might seem simply to be the number of ways in which the elements of the base set may be combined. But this is not quite correct. Each new abstraction identifies a new phenomenon and thus becomes a potential constituent in a further abstraction. The

process, then, is inherently recursive. The analysis should take account of this.

Also of importance is the fact that there are two quite different ways in which identifications may be combined to form a new abstraction. First, there is the process of *composition* in which identified phenomena are combined together as parts to form a new whole. Second, there is the process of *classification* in which identified phenomena are gathered together (as whole elements) into a class of alternatives. (In AI terms, the former is construction using PARTOF relationships and the latter is constructing using ISA relationships.) Every possible subgroup of identifications is a candidate for both processes. Thus, starting from any base set, we may derive a set of abstractions by treating each possible subgroup as (a) a composite and (b) a class.

The general idea is visualised in Figure 1. Here the base set of identifications is labelled P_0. From P_0, we obtain P_1: each identification in this set is an abstraction derived by applying composition or classification to a subset of P_0. Treating P_1 as the base set permits the derivation of a set P_2 in which each phenomenon is the result of classification or composition applied to a subset of P_1. Treating P_2 as the base set permits the derivation of the set P_3 and so on. In this manner, we can go on to derive P_4, P_5, P_6 etc.

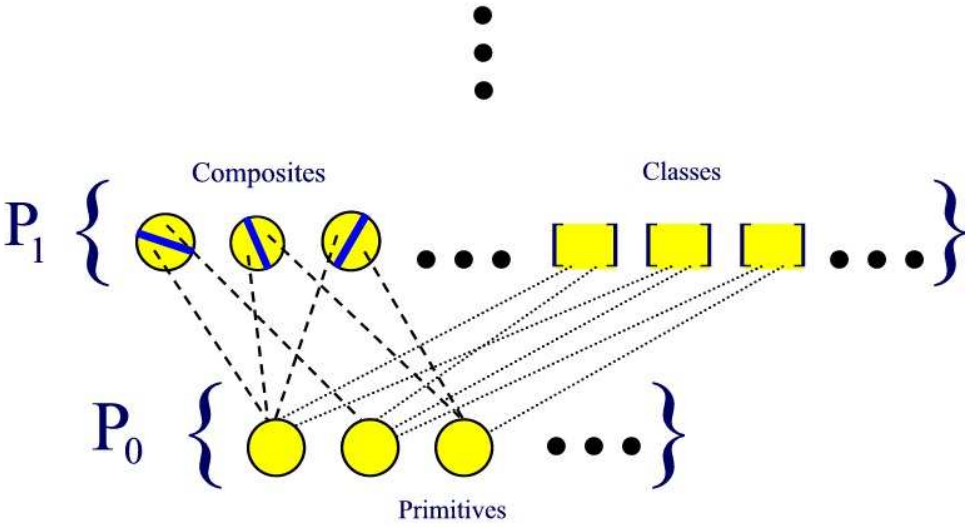

Figure 1: Abstraction tree.

The diagram portrays the generation of possible abstractions, starting from a set of primitives. The dynamics of this are characteristically constructive. But the eliminative aspect should also be apparent. In the case of both composition and classification, a set of elements is reidentified as a single entity. Information relating to the elements themselves is effectively eliminated. But the result is achieved in two different ways. In the case of composition, the elements become the component parts of a new whole. In classification,

the elements become alternative manifestations of a single identity.

3 Complexity

Applied recursively to a base set of identifications, the two forms of abstraction lead to an infinite hierarchy of constructs. The number of nodes in this hierarchy expands rapidly as we move upwards from level to level. Let us say there are n nodes in a particular layer. Then we would expect the number of nodes in the layer above to be

$$2(2^n)$$

since the number of possible combinations of n objects is 2^n, and the process generates two nodes for each combination. However, we must also account for the fact that some of these nodes are redundant. Clustering applied to all possible classes of a set of objects is redundant, since any class obtained must be identical in object membership to one of the original classes. By the same token, abstractions involving classes composed of classes are redundant. Thus we need to discount the nodes which result from classification applied *purely* to classes.

Exactly half of the n nodes will be classes. Therefore we should subtract $2^{\frac{n}{2}}$. The revised formula for the number of nodes then becomes

$$2(2^n) - 2^{\frac{n}{2}}$$

It might seem that a further modification should be made to take account of the fact that exactly n of the 2^n possible combinations are singleton sets, i.e., they simply yield 'copies' of nodes at the layer below. (We might discount these nodes by subtracting $2n$.) However, since it is possible in principle for abstractions to be constructed out of nodes at different levels in the tree, it simplifies matters if we allow the singleton sets to remain. This way, every level of the tree contains a copy of every node at every level below and the possibility of cross-level abstractions is automatically taken into account.

To render the formula in a recursive form is now straightforward. If n_0 is set equal to the number of basic elements, the number of nodes n_i represented at the i'th level of the hierachy may then be calculated using the following recursive formula.

$$n_{i+1} = 2(2^{n_i}) - 2^{\frac{n_i}{2}}$$

4 Significant abstractions

The growth formula reveals the exponential cost of abstraction formulation. But it applies specifically to the case of exhaustive (i.e., unrestrained) abstraction rather than to abstraction in practice. The difference is significant. Agents which form abstractions in a realistic situation are surely unlikely to do so *exhaustively*. More likely, they will aim to ensure that abstractions match up to reality. This will involve making sure that any identifications formed are literally *significant*, i.e., identify real and salient phenomena. (Concerns about the inaccessibility and/or implausbility of objective reality are ignored here.) The net effect is that the set of *significant abstractions* for any particular individual is likely to be a small subset of the total set of possible abstractions.

But although the costs of in-practice abstraction may be lower than the growth formula suggests, they will not be as low as we might hope. They will, after all, include the

costs of carrying out the 'reality check', i.e., whatever operation is required to ensure that abstractions match reality. In the case of classification, this may involve nothing more than making observations about similarities among the relevant class members. But in the case of compositional abstraction, the resulting construct is only valid if the elements fit together in the right way, i.e., only if they have the right relationships. Thus the formation of compositional abstractions always involves the identification, by the abstracting agent, of the relevant relationship. There is evidence to suggest that in the worst case this may be an infinitely complex task (7).

5 Types and tokens

Any abstraction whose structure (in the hierarchy) is not, at any stage, mediated by classification (i.e., whose roots do not go back through any class nodes) has only one, possible grounding in basic elements. In the perception of the agent, there is only one way that it exists. As a conceptualisation, then, the abstraction constitutes a *token*. In contrast, any abstraction whose derivation is mediated by classification (i.e., whose roots do go back through class nodes) identifies a phenomenon with more than one possible grounding in basic elements. With respect to the given set, the latter constitutes a *type*, since it effectively stands for more than one combination of elements.

The theory thus gives a formal meaning to the long-standing distinction between types and tokens. But note how it upgrades the idea from a simple dichotomy into a continuous dimension. As noted, the roots of a type node must go back through one or more class nodes. But there can be more or less of these. And they may appear higher or lower in the tree. Thus, the 'typeness' of a phenomenon is not a black-and-white issue. Rather, it is a matter of (2-dimensional) degree.

How then should we properly render the distinction between types and tokens? A simple approach might be to treat *every* phenomenon as a type, and to say that the 'typeness' of a particular phenomenon is just the size of its extension — the number of ways in which it can exist. A token could then be thought of as a type with an extension of one.

An alternative would be to treat an identification as a type only if its class nodes are sufficiently close to the surface, i.e., appear sufficiently high in the relevant abstraction construct. There might be problems in identifying a suitable cutoff point. But the approach has its attractions. It would, for example, avoid the necessity of treating the individual called Fred Bloggs as a 'type' simply on the grounds that he may consist, at any one time, of quite different arrangements of quantum states. The roots of 'Fred Bloggs' may go back through class nodes, we could argue, but they are at too great a depth to be treated as significant.

Perhaps the best approach is simply to accept that the traditional type/token terminology over-simplifies reality. The logical structure of abstraction means that the size and character of a phenomenon's extension may vary in a range of ways. Therefore there can be no hard and fast distinction between types and tokens.

6 Other applications of the theory

The growth formula and its underlying principles provides a definition of the term 'abstraction' and a means of estimating the number of abstractions which may be formed by an individual agent in a given situation. It provides a *quantitative* theory of abstraction rather than a qualitative one, since it says nothing about what abstractions will actually

consist of, except that they will involve the combination of certain elements. It also says nothing about the way in which the process works or why it is required.

The theory can be applied to natural agents and even, in principle, to human subjects. But here there is always the problem of identifying the set of basic identifications upon which abstraction may build. Without a specification for this set, the growth formula cannot be applied and the rest of the theory becomes inoperable. In some cases, it may be feasible to treat an agent's sensory stimuli as the set of fundamental identifications of phenomena. (Certainly, there could be no more basic set of primitives than this.) But in practice the approach still raises horrific problems of enumeration.

More practical are applications which focus on artificial agents, particularly when these are hand-designed. Very often, the basic set of environmental objects with which a designed agent engages can be simply read-off the design. The derivation of the potential abstraction tree is then straightforward.

In some cases, it may be useful to map out an agent's total abstraction set simply as a means of evaluating its possible, representational trajectories (cf. 8). This might also provide the basis for an evaluation of the agent's conceptual *adventurousness*. Its relative penetration of the total abstraction set — the ratio between the number of developed and potential abstractions — summarises the degree to which the agent has fleshed-out the potential conceptualisations of its environment. Relative penetration might then become a kind of conceptual 'horse-power' rating for artificial agents. (This is not completely satisfactory, however, since the total abstraction set will normally be significantly larger than the set of significant abstractions.)

7 Representation and behaviour

To some degree, the theory may also be used to make judgements about the representational behaviour of agents. An agent's total abstraction set includes all the phenomena that the agent is capable of identifying (including ones that do not actually exist). Putting this in representational terms, we would say that the abstraction tree identifies the complete set of phenomena that the agent is capable of representing, as well as the logical dependencies between them.

Thus, if a particular phenomenon does not appear within an agent's total abstraction set, we know that the agent *cannot* form a representation for that phenomenon. It may be unable to form a particular representation for other reasons. But the absence of the phenomenon from the abstraction set shows that it cannot do so *in principle*. This might become an issue of importance, for instance, if an attempt were being made to build an agent that would acquire the ability to behave contingently with respect to a property of the world that it was unable to represent.

Imagine for example that the aim is to construct a simple, mobile agent which will acquire the ability to approach smooth objects but not spikey ones. Regardless of any efforts made, the experiment will necessarily fail if the phenomenon 'smooth object' has no representation within the agent's abstraction tree.

But the representational implications of the theory only go so far. It allows one to calculate what is contained within a particular abstraction set and thus what a particular agent is and is not capable of representing. However, it says nothing about what a particular representation will consist of or how it will be constructed. (This is really just the same point as was made above: the theory does not specify what an abstraction will consist of, merely that it must combine certain elements.)

Furthermore, no claim is made that agents will representationally reproduce the *structure* of abstraction trees. Indeed, it is apparent that areas of research interested in learning and behaviour acquisition (whether motivated by a representational interests or not) show little sign of devising methods which generate abstraction trees, or anything like them. If anything, the reverse is the case (cf. 9). The evidence is that insofar as contemporary artificial agents may be said to build representations *at all*, these do not resemble abstraction trees.

On the other hand, it is noticeable that the very same areas of research tend to divide attention between classification (class-forming) methods and compositional methods. In other words, they may be viewed as dividing attention between the two fundamental processes of abstraction. This is perhaps most noticeable in machine learning, which is broadly divided up into a subfield focussing on statistical classification methods (similarity-based learning) and a subfield focussing on compositional or relational methods (discovery, analogy, inductive logic programming etc.) (10)

8 Explicit and implicit learning

Perhaps the most fruitful area for applications of the theory is that of cognitive psychology. A lively debate in this area involves the problem of explicit v. implicit learning. In part, this is concerned with the question of whether knowledge is stored in an abstract or specific (i.e., instance-based) form. It also focusses on the degree to which knowledge is the result of conscious or unconscious processes. (In some sense, the two parts of the debate are really one, with the former focussing on static issues and the latter on dynamic.)

Traditionally, the implicit/explicit issue has been researched using experiments in which human subjects are either taught, or exposed to strings generated by an artificial grammar. By evaluating the subject's ability to classify test cases, or to transfer knowledge from one grammar to another, deductions are attempted showing the degree to which abstractions have or have not resulted from implicit learning processes.

The seminal work in this area was performed by Reber (11) and it was his main conclusion that unconscious (i.e., implicit) processes of learning could produce internal abstractions with the same functional properties as those acquired through explicit tuition. The implications that Reber drew from his results have been widely questioned, with objections often focussing on the fuzziness of the supposed dichotomy between abstract and specific knowledge.

All business as usual, perhaps. But from the point of view of quantitative abstraction theory, it begins to look as if the problem here, as with the dilapidated type/token distinction, may really be the result of the attempt to apply an over-simplified, black-and-white conceptualisation to what is in reality a complicated, multi-dimensionsal issue.

According to the theory, abstractions have a logical structure which may be *arbitrarily* deep. The derivation of new abstractions, whether classificatory or compositional, may proceed at any level within the tree of existing abstractions. On this view, it makes no sense to classify new knowledge as either abstract or specific. Rather, it should be identified as having a particular *level* of abstraction.

Similar remarks can be made with respect to conscious v. unconscious processing. Assuming that the level of 'consciousness' inherent in cognitive processing is a function of the abstractness of the entities over which it is applied, we can apply the continuity upgrade to the conscious/unconscious 'dichotomy' too. On the basis of the assumption stated, we can treat the issue of the consciousness of processing as a matter of degree, and

judge any specific cases according to elevation in the relevant abstraction tree.

9 Concluding comment

It would be an interesting project to determine how many of the unresolved issues surrounding the question of implicit v. explicit learning would evaporate in the presence of suitable enhancements in the terminology. The project would certainly be approved by Alan Newell who nearly forty years ago admonished psychologists for their use of simplistic, binary opposites in their conceptualisation of cognitive function (12) Rather obviously, Newell's criticisms have had a limited impact. In fact, the implicit v. explicit debate gained its principal momentum a full decade *after* Newell's publication. But the fact that cognitive psychologists still adhere to black-and-white concepts may be due to the fact that workable replacements have yet to be provided. In this context, the limited but concrete contribution of the present theory may have a worthwhile future.

References

[1] Aristotle, (1984). In J. Barnes (Ed.), *The Complete Works of Aristotle*. Princeton: Princeton University Press.

[2] Locke, J. (1690/1978). In A. Pringle-Pattison (Ed.), *An Essay concerning Human Understanding* (first pub. 1690). Sussex: Harvester.

[3] Plaisted, D. (1981). Theorem proving with abstraction. *Artificial Intelligence*, *16* (pp. 47-108). 1.

[4] Sacerdoti, E. (1974). Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, *5* (pp. 115-135). 2.

[5] Knoblock, C. (1994). Automatically generating abstractions for planning. *Artificial Intelligence*, *68*, No. 2.

[6] Hume, D. (1740). *A Treatise of Human Nature* (second edition). Oxford University Press.

[7] Clark, A. and Thornton, C. (1997). Trading spaces: computation, representation and the limits of uninformed learning. *Behaviour and Brain Sciences*, *20* (pp. 57-90). Cambridge University Press.

[8] Clark, A. and Karmiloff-Smith, A. (1993). The cognizer's innards: a psychological and philosophical perspective on the development of thought. *Mind and Language*, *8*.

[9] Brooks, R. (1991). Intelligence without reason. *Proceedings of the Twelth International Joint Conference on Artificial Intelligence* (pp. 569-595). San Mateo, California: Morgan Kaufman.

[10] Russell, S. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach* (Second Edition). Prentice Hall.

[11] Reber, A. (1967). Implicit learning of artificial grammars. *Journal of Verbal Learning and Verbal Behaviour*, *5* (pp. 855-863).

[12] Newell, A. (1965). Limitations of the current stock of ideas for problem solving. In A. Kent and O. Taulbee (Eds.), *Conference on Electronic Information Handling*. Washington, D.C.: Spartan.

AISB Journal **1**(3), © SSAISB, 2003
http://www.aisb.org.uk

Cindy: A 3D Virtual Entertainer

Fabio Zambetta, Graziano Catucci and Fabio Abbattista

V.AL.I.S. Group
Dipartimento di Informatica, Università degli Studi di Bari
Via E. Orabona 4, Bari, I-70124
zambetta@di.uniba.it ; gracat@email.it ; fabio@di.uniba.it

Abstract

Intelligent virtual agents exhibiting autonomous behavior rather than mere reactions to users actions are going to become a major requirement for modern web sites. In this paper we present Cindy, a 3D agent character designed with SAMIR, a system conceived to create intelligent agents with a 3D animated look as a front-end, to enhance the user interaction with the web applications it's embedded into. Our system relies on an XCS classifier system to achieve an autonomous behavior: New rules are generated during the interaction with the users by the XCS discovery component, resulting in novel behavioral patterns.

1 Introduction

Intelligent personal agents are software components designed to advise web applications users, where a high level of human computer interaction is required. Indeed their aim is to substitute the classical WYSIWYG interfaces, which are often difficult to manage by casual users, with reactive and possibly pro-active virtual ciceros able to understand users wishes and converse with them, find information and execute non-trivial tasks usually activated by buttons pressing and menu choices. Frequently these systems are coupled with an animated 2D/3D look-and-feel, embodying their intelligence via a face or an entire body. This way it's possible to enhance users trust into these systems simulating a face-to- face dialogue as reported in (Cassell et al., 2000).

A very complete agent of this kind, frequently called an ECA (Embodied Conversational Agent), is REA (Cassell et al., 2000), a Real Estate Agent able to converse with the users and sell them a house with regard to their wishes and needs. The interaction occurs in real time via sensor acquiring user facial expressions and hands pointing; moreover speech recognition is performed to avoid users type their requests. REA answers using its body posture, its facial expressions and digitized sounds rendering the salesperson recommendations and utterances. The EMBASSI system (Jalali-Sohi and Baskaya, 2001) was born by a very big consortium occupied in defining the technologies and their ergonomics requirements to implement an intelligent shop assistant to facilitate user purchasing and information retrieval. These objectives are pursued by multi-modal interaction: common text dialogs as well as speech recognition devices are used to sense user requests whilst a 3D face is the front-end of the system. The agent is able to send its response also via classical multimedia content (video-clips, hyperlinks, etc.). In (Kshirsagar and Magnenat-Thalmann, 2002) an agent is described which is not just able to be animated

but also to answer to users based on emotions modeled on the Five Factor Model (FFM) of personality (McCrae and John, 1992) and implemented using Bayesian Belief Networks.

Moreover the Alice chatterbot is used in order to let the web agent process and generate responses into the classical textual form.

The SAMIR system was created in order to obtain a lighter system than REA, usable from virtually every browser and operating system, easily embeddable into web applications as EMBASSI and exhibiting an autonomous behavior, thanks to the XCS algorithm, even more than the various Bayesian Belief Networks solutions such as the one proposed in (Kshirsagar and Magnenat-Thalmann, 2002).

Moreover we are spending many efforts in the creation of a custom 3D faces editor in order to let the users personalize, trough a simple web application accessible via a web browser, any face used by SAMIR: This is a clear advantage of our system with respect to the cited ones because none of them integrates a similar built-in component.

The SAMIR system was used to create a "family of prototypes": The first one named Samir was our first attempt to create a working 3D agent, but it suffered from a simply implemented behavior because the XCS-based module was not still fully implemented.

While enhancing our XCS module for Samir, we conceived Cindy, a 3D virtual entertainer with a working intelligent behavior and a quite extended knowledge base in order to mimic a girl very devoted to gossips and chattering: This gave us cues and suggestions, matured during a set of experiments with end users, about how users perceived Cindy and about how we could designing Cindy's dialogue structure, to involve users in interesting conversations. Finally we are currently implementing UPoet, the prototype of a "virtual poet". Our prototype uses the generative module YODA devoted to create haikus (a minimalist form of Japanese poetry) (Higginson, 1995): It plays and displays small poetries, keeping his facial expressions and conversational tone coherent with its read verses.

The remainder of this paper is organized as follows: the next section depicts the overall architecture of the SAMIR system while Section 3 gets into details of the proposed chatterbot, called Cindy. Section 4 addresses experimental results and finally, Section 5 draws out future work directions.

2 Overview of the System

SAMIR (see Figure 1) is a client-server system, composed of 3 main sub-systems detailed in the next sections: the **Dialogue Management System (DMS)**, the **Behavior Manager** and the **Animation System**.

The DMS is responsible for directing the flow of information in our system: When the user issues a request from the web site, via a form embedded into the HMTL page displayed by the web browser, an HTTP request is directed to the DMS Server to obtain the HTTP response storing the chatterbot answer. At the same time, based on the events raised by the user on the web site (such as clicking specific links, changing the current web page, selecting particular menus and/or buttons) and on his/her requests, a communication between the DMS and the Behavior Manager is set up. This results into a string encoding the expression the Animation System should assume. This string specifies coefficients for each of the possible morph targets (Fleming and Dobbs, 1998) into our system: We use some high-level morph targets corresponding to the well-known fundamental expressions (Ekman, 1982) but even low-level ones are a feasible choice in order to preserve full MPEG-4 compliance. After this interpretation step, a key-frame interpolation is performed to animate the current facial expression.

The SAMIR system allows us to easily create intelligent agents with a 3D animated

look as a front-end, to enhance the user interaction with the web applications it is embedded into (Abbattista et al., 2002).

In (Zambetta et al., 2003), as an example, SAMIR has been used to create a virtual book-seller, able to support users searching for a book into a Web book-store.

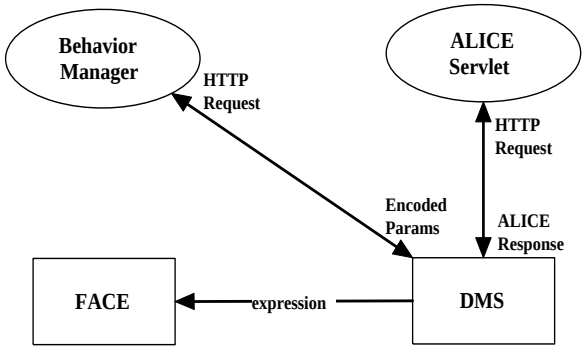

Figure 1: The overall architecture of SAMIR

2.1 The Animation System

The FACE (Facial Animation Compact Engine) System is an evolution of the Fanky Animation System (Zambetta et al., 2001). Fanky was developed to overcome some limitations of the Tinky system (Paradiso et al., 1999), namely the technologic bottleneck due to the EAI (External Authoring Interface) API. It did not allow any application to reach high framerates, whilst using Shout3D API applets ensure good performances and even a very good OpenGL based hardware support, also with respect to "advanced" rendering features such as, for example, bump mapping, environmental mapping and multi-texturing. Besides, Shout 3D is almost 100% VRML '97 (The VRML '97 Specification) compatible and it has some built-in applet resembling the basic Open Inventor viewer classes, very useful to fasten the application development.

FACE design resembles Fanky in the implementation of SACs (Standard Anatomic Components). The basic idea underlying them is to define face regions, acting as objects, in an object-oriented sense of the term. The offered services correspond to different low-level deformations such as FAPs (Facial Animation Parameters), used during the animation process. The **Facial Animation Parameters (FAPs)** are the set of parameters defined by MPEG-4 to allow the animation of synthetic face models (The MPEG-4 standard specification) . They are divided into a high level set, used to describe visemes and expressions, and a low level set, describing facial deformations in terms of FDPs (Facial Definition Parameters), a minimum collection of "fundamental" points.

SACs may offer even services such as face sculpting and remodeling, used when the user wants to modify the 3D face mesh, as detailed later on in this section when we sketch our custom editor out. Finally, using SACs we could choice the numerical method employed to deform vertices associated to a particular region of the 3D face, at runtime.

After having employed different numerical methods in Fanky (i.e. FFD (Sederburg, 1986), Waters muscle model (Waters, 1987) and various key-frame interpolation schemes (Bergeron, 1986)), we choose to use the linear interpolation of a 3D face key-frames, as exported by the 3D Studio Max morph targets.

Our choice was dictated by the experimental evidence that, this method represents in our opinion, the best compromise between speed and accuracy.

We used a Shout 3D plug-in to export morph targets data straight from the 3D Studio Max 4.0 modeler to the S3D file format, considerably shortening the setup process required for integrating a new face into the SAMIR system. Indeed, the morph targets meshes were generated using the well-known FaceGen software (The FaceGen website): This enabled us to bypass a "preprocessing" phase usually devoted to the manual creation of a detailed 3D face, saving some other time.

FACE supports a variable number of morph targets: For example we currently use either 12 high-level ones or the number of the entire "low-level" FAP set, in order to achieve MPEG-4 compliance (The MPEG-4 standard specification) .

In the high-level parameters scenario we are just interested to manipulate a facial expression as a combination of "fundamental" expressions, reported in table 1, a quite simple representation to be used in conjunction with our XCS Classifier System in the Behavior Generator, the module driving Cindy's behavior.

In fact each effector of the XCS matches exactly one expression, but we need exactly 4 bits to encode every single expression (see section 2.3 for more details about that).

Even in this case we have the effector part of each XCS rule is 48 (equal to 12*4) bits long, which is always enough to create storage and debugging problems when the set of rules starts to considerably grow.

Using just a small set of high-level parameters might be extremely useful when trying to avoid bandwidth limitations, a major advantage in porting this animation module to a small device such as a Pocket PC (The Pocket PC website), a process we are beginning to experiment with.

Table 1: The set of high-level morph targets.

Expression	**Type**
Anger	Stimulus-response
Disgust	Stimulus-response
Fear	Stimulus-response
Sadness	Stimulus-response
Happiness	Stimulus-response
Surprise	Non-conscious reflex
Left Blink	Non-conscious reflex
Right Blink	Non-conscious reflex
Left Brow Up	Non-conscious reflex
Right Brow Up	Non-conscious reflex
Left Brow Down	Non-conscious reflex

In the low-level scenario, on the contrary, we use 68 morph targets, one for each allowed FAP: This maps badly to be used with the Behavior Generator because the length of the rule will be overwhelming and hard to handle.

This option is provided if we want Cindy to be employed in very simple applications where a fixed logic rule set might turn out to be a useful choice, eliminating the need of the XCS.

A possible scenario of this kind is represented by the use of Cindy as a simple virtual presenter or speaker on web sites, web radios etc.

It is worth noting that an unlimited number of timelines can be used for animations in FACE, allocating one channel for the stimulus-response expressions, another one for

eye-lid non-conscious reflexes, another one for head non-conscious reflexes and so on.

We are currently developing a custom editor able to perform the same tasks performed by FaceGen but optionally giving more control to the user: This way, we believe, each user, both the unexperienced one and the experienced one, might enjoy the process of creating a new face, tailored to his/her wishes, who could use some specific low-level deformation tools, based upon the well-known FFD technique (Sederburg, 1986).

2.2 The Dialogue Management System

The **DMS** (Dialogue Management System) is responsible for the management of user dialogues and for the extraction of the necessary information for giving textual responses to any user. The **DMS** can be viewed as a client-server application composed mainly by two software modules, communicating through the HTTP protocol (see Figure 2). The **client side** application is just a simple Java applet whose main aim is to let user to type requests in a human-like language and to send these ones to the **server side** application in order to process them.

The other important task it is able to perform is retrieving specific information, based on the responses elaborated by the server-side application, on the World Wide Web through the JavaScript technology. On the **server side** we have the **ALICE** Server Engine enclosing all the knowledge and the core system services to process user input. ALICE is an open source chatterbot developed by the ALICE AI Foundation and based on the **AIML** language (Artificial Intelligence Markup Language), an XML-compliant language that gives us the opportunity to exchange dialogues data through the World Wide Web. The AIML knowledge base of the bot is composed of several categories. Here is an example of two categories in AIML:

```
<category>
<pattern>*</pattern>
<template> Hello! </template>
</category>

<category>
<pattern>I WOULD NOT *</pattern>
<template> No one says you have to. </template>
</category>
```

The <category> tag indicates an AIML category, the basic unit of chat robot knowledge.

The category has a <pattern> and a <template>. The pattern in the first example is the wildcard symbol '*' that matches any input. In general the pattern is a combination of text and wildcards as in the second example ("I WOULD NOT *"). The template is the bot's answer associated to the user input detected by the pattern. Just the text "Hello!" in the first example or "No one says you have to." in the second example.

ALICE has been fully integrated in SAMIR as a Java Servlet and all the knowledge of the system has been stored in the AIML files, containing all the patterns matching user input.

Dialogues data are exchanged through simple built-in classes handling the classical HTTP sockets communication.

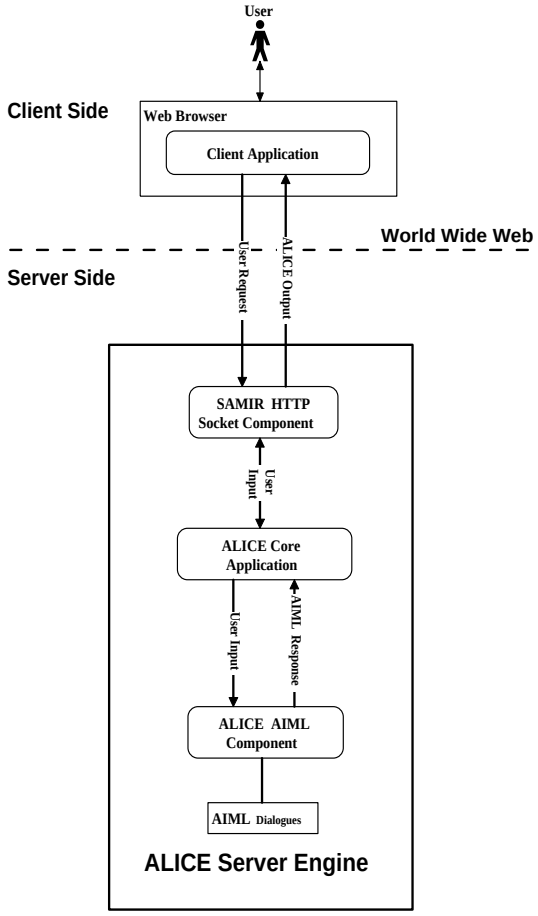

Figure 2: The DMS Architecture

2.3 The Behavior Generator

The Behavior Generator aims at managing the consistency between the facial expression of the character and the conversation tone. The module is mainly based on Learning Classifier Systems (LCS), a machine learning paradigm introduced by Holland in 1976 (Holland, 1976). The learning module of SAMIR has been implemented through an XCS (Wilson, 1995), a new kind of LCS, which differs in many aspects from the traditional Holland's framework. The most appealing characteristic of this system is that it is closely related to the Q-learning but it can generate task representations which can be more compact than tabular Q-learning (Watkins, 1989). At discrete time intervals, whose duration depend on the particular agent implementation, the agent observes a state of the environment, takes an action, observes a new state and finally receives an immediate reward.

The basic components of an XCS are:

- **Performance Component**, that, on the ground of the detected state of the environment, selects the better action to be performed.

- **Reinforcement Component**, whose aim is to evaluate the reward to be assigned to

the system.

- **Discovery Component** which, in case of degrading performance, is devoted to the evolution of new, more performing rules.

The environment in which SAMIR has to act is represented by the user dialogue (the higher the user satisfaction the higher the reward received from SAMIR). At the very beginning of its life, the behavior of SAMIR is controlled by a set of random generated rules and, consequently, its capability is very low.

Behavior rules are expressed in the classical format **if** <condition> **then** <action>, where <condition> (the state of the environment) represents a combination of 4 possible events, sensed by 4 effectors, representing different conversation tones such as: user salutation (user performs/does not perform salutation), user request formulation to the agent (no request, polite, impolite), user compliments/insults to the agent (no compliment, a compliment, an insult, a foul language), user permanence in the Web page (user changes/does not change the page) while <action> represents the expression that the Animation System displays during user interaction. In particular, the expression is built as a linear combination of a set of fundamental expressions that includes the basic emotion set proposed by Paul Ekman, namely anger, fear, disgust, sadness, joy, and surprise (Ekman, 1982). Other emotions and many combinations of emotions have been studied but remain unconfirmed as universally distinguishable. However, we have extended the basic set of expressions in order to include some typical human expression such as bother, disappointment and satisfaction. The Behavior Manager, as explained above, is able to produce synthetic facial expressions, to be shown according to the content of the ongoing conversation. Thus the <action> part provides the Animation System with the percentage of each one of the expressions, to be used to compose the desired expression of our character. For example, an expression composed by 40% of joy and 60% of surprise is coded into the string reported in Table 2.

It should be noted however that we encode the value of the expressions in 10% steps, so that **0100** (i.e. the integer number 4) stands for 40%, **0110** stands for 60% and so on.

Table 2: Encoded String.

0100	**0000**	**0110**	**0000**	**0000**	**0000**	**0000**	**0000**	**0000**
% of Surprise	% of Sadness	% of Joy	% of Fear	% of Disgust	% of Anger	% of Bother	% of Disapp.	% of Satisfaction

3 Cindy, the virtual entertainer

The framework provided by SAMIR allowed us to create a 3D agent able to entertain users by chatting with them on several different topics (from religion to politics or gossip). When the user connects to Cindy (http://ant1.di.uniba.it:8080/SamirBeta2/), she welcomes the user and waits for an input from him. The user, at this point, can start a conversation concerning every topic he wants to discuss with Cindy. Cindy, contrarily to the virtual book-seller implemented in (Zambetta et al., 2003), does not perform any useful task and her aim is only to converse with the user.

Our goal in the experimentation was twofold: i) to test the effectiveness of the knowledge (AIML files) provided with the ALICE package and, ii) to test the quality of the

synchronization between the character expressions, produced by the animation system (FACE) and the dialogues managed by the DMS module.

A first experiment was performed in order to verify the capability of the DMS module to dialogue with users (in this experiment no 3D character has been used). A sample of 10 users was selected, with different skills and knowledge about our system. Users were asked to interact with Cindy during an entire week, and to report their comments and suggestions.

During the first 3 days, users were free to choose the topics of the conversation, in order to get acquainted with the system. In the second part of the week, the topics of the conversation were imposed (*Computer Science, Gossip and Sports, Politics and Religion)*.

All the interactions have been recorded in order to analyze quantitative and qualitative factors. Over the period of the experimentation, 1,096 dialogues were recorded (an average of 100 dialogues for each user). The dialogues length ranges from 20 minutes to 293 minutes and each user spent, on average, 70 minutes per day interacting with Cindy.

The mean length of the dialogues was 126 minutes over the free conversation period and 108 minutes over the guided conversation period.

From the analysis of the log files we noticed several situations in which Cindy was not able to keep the dialogue going with the users. We classified these situations in 6 different categories. Table 3 shows the frequency observed in the logs with respect to each provided category. At the end of the experiment, users were asked to report their own opinions about their dialogues with Cindy (see Table 4).

Table 3: Frequency of the most common errors.

Category	**Frequency**
Quoted Question	29%
Fall-back	20%
Misunderstood Question	16%
Wrong Answer	15%
Evasive Answer	11%
Unknown topic	6%

Table 4: User evaluation of the dialogues.

Evaluation	**Frequency**
Interesting	80%
Funny	80%
Ambiguous	80%
Intelligent	60%
Involving	40%
Sensible	40%
Real	30%
Up-to-date	30%

From the analysis of the separate dialogue logs, both positive and negative characteristics emerged.

- **Positive characteristics**. In some dialogues, the Event Interpreter module exhibits a sort of ironic behavior. In another situation the module has been able to defend

itself from the critics of a user. In several situations the chatterbot gave some nonsense answer but, unexpectedly, users found it very nice and were attracted by it.

- **Negative characteristics**. The chatterbot is not able to manage indefinite pronouns. In very complex dialogues concerning many different interrelated topics, the chatterbot was not able to continue the dialogue. This situation occurred more frequently when dialogues were on very specific topics like *Science, Religion or Sport*.

Motivated by the quite positive results, we performed a second experiment whose aim was to test both the reactions of users in front of a 3D character and the effectiveness of our animation system. This time, the same ten users of the first experiment, were asked to freely converse with Cindy and to report their impressions about the 3D interface (Tables 5 and 6).

Table 5: User evaluation of the facial expressions.

	Not at all	**Very few**	**Few**	**Don't care**	**Sufficient**	**Quite good**	**Very good**
Human-like		17%			50%	33%	
Sensible				17%	33%	50%	
Useful		17%		17%	33%	16.5%	16.5%
Coherent			17%		33%	33%	17%
Pleasing			17%		50%	17%	16%
Funny				17%	33%	33%	17%
Distracting	50%		33%	17%			

Table 6: User evaluation of the overall interaction.

	Not at all	**Very few**	**Few**	**Don't care**	**Sufficient**	**Quite good**	**Very good**
Polite					33%	17%	50%
Efficient		17%	17%	16%	50%		
Simple				17%	17%		66%
Proficient			17%	33%	33%		17%
Pleasing				17%	50%	17%	16%
Exciting	17%		17%	33%		17%	16%

From the analysis of the separate dialogue logs and from the collected statistics, we devised that most of the users considered Cindy very simple and funny. However, many users complained about the fact that the chatterbot is not able to internally store the current context of dialogues. Some users would have preferred to have the possibility to modify the look of the character on their taste (in particular, male users would have preferred to change some aesthetic features of Cindy).

Some typical situations occurred in this experiment have been: i) users insulted Cindy (see Fig. 3), in this case Cindy had a disappointed expression; ii) users were very kind with Cindy (see Fig. 4), in this case Cindy had a joyful expression.

User> What are you?
Cindy> I am the latest result in artificial intelligence, which can reproduce the capabilities of the human brain with greater speed and accuracy.
User> You are stupid.
Cindy> (disappointed) I will remember you said that when robots take over the World.

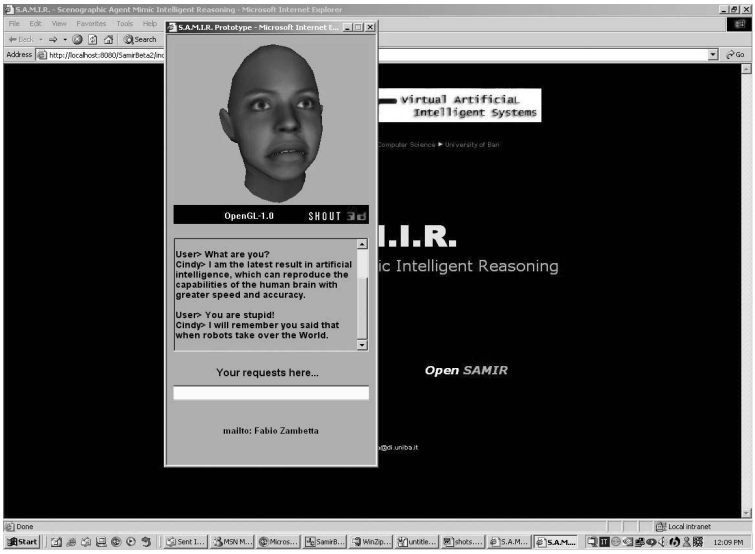

Figure 3: A user insults Cindy

User> I'm nice.
Cindy> Let me be the judge of that.
User> Am I nice?
Cindy> You seem nice to me.
User> I would like you to know that you are very nice.
Cindy> (happy) Ok, is that a fact. Thanks, I like you too.

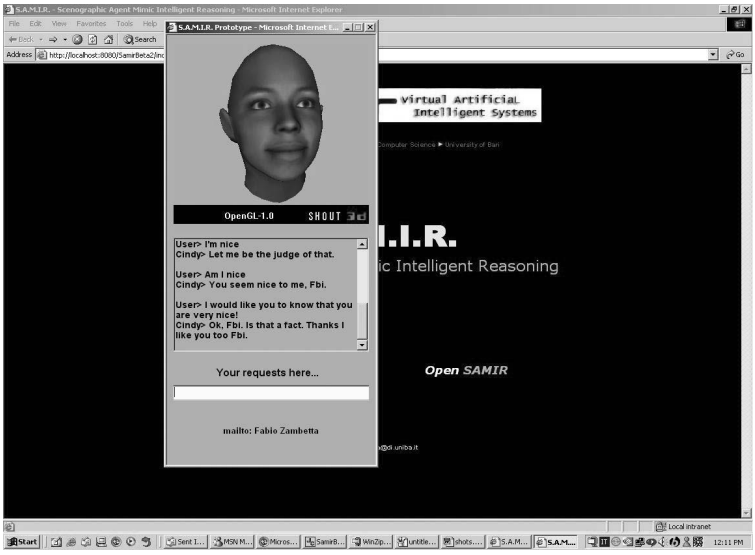

Figure 4: A user is very nice with Cindy

4 Conclusions and Future work

In this paper we presented a first prototype of a 3D agent able to entertain users by chatting with them on several topics. The preliminary experiments performed so far showed that users were quite satisfied of the 3D agent and, in several situations, they were amused by Cindy's non sense answers.

Even if we can rely on a well performing prototype, our work will be aimed to give a more natural behavior to our agent. This can be achieved improving dialogues, and eventually, the text processing capabilities of the ALICE chatterbot, and giving Cindy a full proactive behavior: the XCS should be able not only to learn new rules to generate facial expressions but also to modify dialogue rules, to suggest interesting links and to supply an effective help during some site navigation.

Moreover, Cindy has been built up to be used as working prototype by which we intend to improve the capability of the DMS in order to address issues concerning both multi- lingual interactions and text-to-speech technologies.

References

Abbattista, F. Paradiso, A., Semeraro, G., and Zambetta, F. (2002). An agent that learns to support users of a web site. In Roy, R. Koeppen, M., Ovaska, S., Furuhashi, T., and F., H., editors, *Soft Computing and Industry: Recent Applications*, pages 489–496. Springer.

Bergeron, P. (1986). Techniques for animating characters. *SIGGRAPH Course Notes*, 22:240–265.

Cassell, J. Sullivan, J., Prevost, S., and Churchill, E., editors (2000). *Embodied Conversational Agents*. MIT Press, Cambridge.

Ekman, P. (1982). *Emotion in the human face*. Cambridge University Press, Cambridge.

Fleming, B. and Dobbs, D. (1998). *Animating Facial Features and Expressions*. Charles River Media, Hingham.

Higginson, W. J. (1995). *The Haiku Handbook*. Kodansha Europe, London.

Holland, J. H. (1976). Adaptation. In Rosen, R. and Snell, F. M., editors, *Progress in theoretical biology*. Plenum, New York.

Jalali-Sohi, M. and Baskaya, F. (2001). A multimodal shopping assistant for home e-commerce. In Rosen, R. and Snell, F. M., editors, *Proceedings of the 14th Int'l FLAIRS Conf.*, pages 2–6. Key West.

Kshirsagar, S. and Magnenat-Thalmann, N. (2002). Virtual humans personified. In *Proceedings of the Autonomous Agents Conference (AAMAS)*. Bologna.

McCrae, R. R. and John, O. P. (1992). An introduction to the five factor model and its applications. *J. of Personality 60*, pages 175–215.

MPEG-4 standard specification. http://mpeg.telecomitalialab.com/standards/mpeg-4/mpeg-4.htm.

Paradiso, A. Nack, F., Fries, G., and Schuhmacher, K. (1999). The design of expressive cartoons for the web - tinky. In *Proceedings of ICMCS Conference, Florence*, pages 276–281. IEEE Press.

Sederburg, T. (1986). Free form deformation of solid geometric models. In *Proceedings of SIGGRAPH '86*, volume 20 of *Computer Graphics*, pages 151–160, Dallas.

The FaceGen website. http://www.facegen.com.

The Pocket PC website. http://www.microsoft.com/mobile/pocketpc/default.asp.

The VRML '97 Specification. http://vrml.org/technicalinfo/specifications/vrml97/index.htm.

Waters, K. (1987). A muscle model for animating three-dimensional facial expression. In *Proceedings of SIGGRAPH '87*, volume 21 of *Computer Graphics*, pages 17–24, Anaheim.

Watkins, C. J. C. H. (1989). *Learning from delayed rewards*. PhD thesis, University of Cambridge, Psychology Department.

Wilson, S. (1995). Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2):149–175.

Zambetta, F. Paradiso, A., Abbattista, F., and Semeraro, G. (2001). An agent to personalize user interactions. In Rozic, N. Begusic, D., editor, *SoftCOM 2001 - Int. Conf. on Software, Telecommunications and Computer Networks*, pages 431–438. FESB, Split.

Zambetta, F. Catucci, G., Abbattista, F., and Semeraro, G. (2003). Samir: Your 3d virtual bookseller. In *paper submitted to Web3D 2003 Symposium, 8th International Conference on 3D Web Technology*.