

Contents

Preface	ii
<i>Amílcar Cardoso, University of Coimbra, Portugal</i>	
<i>Geraint A. Wiggins, City University, London, UK</i>	
Supporting Creativity	1
Multiagent Systems for Supporting and Representing Social Creativity in Science	3
<i>Francesco Amigoni, Viola Schiaffonati & Marco Somalvico, Politecnico di Milano, Italy</i>	
Supporting Creativity in Software Design	13
<i>Paulo Gomes, Francisco C. Pereira, Nuno Seco, Jos L. Ferreira, Carlos Bento, University of Coimbra, Portugal</i>	
Thinking Through Doing. External Representations in Abductive Reasoning	21
<i>Lorenzo Magnani, University of Pavia, Italy & Georgia Institute of Technology, USA</i>	
Intelligent Dynamic Collaboration	29
<i>Kirsty A. Beilharz, University of Sydney, Australia</i>	
Pattern and Structure in Creative Systems	39
Giving Colour to Images	41
<i>Penousal Machado, Instituto Superior de Engenharia de Coimbra & University of Coimbra, Portugal</i>	
<i>André Dias, Nuno Duarte, Amílcar Cardoso, University of Coimbra, Portugal;</i>	
Exact and Approximate Distributed Matching Problem for Musical Melodic Recognition	49
<i>Costas S. Iliopoulos & Masahiro Kurokawa, King's College, London, UK & Curtin University of Technology, Australia</i>	
A psycho-social model for the evolution of aesthetic patterns	57
<i>Thibaud de Souza & Tatiana Kalganova, Brunel University, UK</i>	
Evolution of Musical Motifs in Polyphonic Passages	67
<i>Costas S. Iliopoulos, King's College, London, UK & Curtin University of Technology, Australia</i>	
<i>Kjell Lemström, University of Helsinki, Finland</i>	
<i>Mohammed Niyad, King's College, London</i>	
<i>Yoan J. Pinzón, Universidad Autonoma de Bucaramanga, Colombia</i>	
Creative Language and Context Generation	77
Linguistic creativity at different levels of decision in sentence production	79
<i>Pablo Gervas, Universidad Complutense de Madrid, Spain</i>	
Awaiting the sensation of a short, sharp shock: twist-centred story generation	89
<i>John Plattsl & Christian Huyck, Middlesex University, UK</i>	
<i>Ann Blandford, UCL Interaction Centre, UK</i>	
Automatic Puzzle Generation	99
<i>Simon Colton, University of Edinburgh, UK</i>	

Preface

This document is the proceedings of the fourth AISB symposium on creativity in AI and Cognitive Science. The purpose of the symposium is to bring together researchers interested in all AI and cognitive aspects of creativity and cultural enterprise. The aim of holding one unified meeting, instead of several simultaneous smaller ones, is to promote communication between those studying different aspects of creativity.

Each workshop has naturally taken a different structure from the last, and this year is no exception. The papers in this volume can best be divided into three areas: Supporting Creativity, Pattern and Structure in Creative Systems, and Creative Language and Context Generation.

The first of these brings together papers which focus on the support of (mostly scientific) creative reasoning by computers.

The second section looks at computational and aesthetic issues in dealing with structure, pattern and repetition in a creative context, both in terms of creative technology and of its applications. This section contains papers on music and visual art.

Finally, the third section is about creativity in a particular domain – language – and creation of context for language generation. The papers in this section cover many different levels of creativity, from the highest level of narrative down to the choice of syntax in a poetic context.

The papers presented here were carefully selected by multiple blind anonymous peer review. This, as always, entailed a lot of work for the symposium's programme committee, to whom we are very grateful. They were:

Simon Colton, Universities of Edinburgh and York, UK

Werner Dubitzky, University of Ulster, UK

John Gero, University of Sydney, Australia

Ashok Goel, Georgia Institute of Technology, USA

Alison Pease, University of Edinburgh, UK

Graeme Ritchie, University of Edinburgh, UK

We are grateful also to the authors for their punctuality and cooperation in preparing this volume, and to the convention organisers for making the event possible.

Amilcar Cardoso

University of Coimbra, Portugal

Geraint A. Wiggins

City University, London, UK

I – Supporting Creativity

MULTIAGENT SYSTEMS FOR SUPPORTING AND REPRESENTING SOCIAL CREATIVITY IN SCIENCE

Francesco Amigoni; Viola Schiaffonati; Marco Somalvico

Politecnico di Milano – Artificial Intelligence and Robotics Project

Dipartimento di Elettronica e Informazione; Politecnico di Milano; Piazza Leonardo da Vinci 32; 20133 Milano; Italy

amigoni@elet.polimi.it; schiaffo@fusb.eta.polimi.it; somalvic@elet.polimi.it

Abstract

In order to address the topic of creativity in science, this paper evidences the two main features that characterize the current scientific practice: the increasingly important role of information machines and of collective sociality. Given the nature of the scientific enterprise, we propose to adopt a powerful and flexible multiagent system, called *scientific social agency*, both to support creative scientists in their work and to represent the models devised as result of their activity. We ground the discussion on two practical relevant examples: the Human Genome Project and the Screensaver Lifesaver Project.

1 Introduction

Creativity is usually considered as a mostly inexplicable human activity characterized by different forms and manifestations. Several attempts to enlighten creative activities and creative behaviors have been proposed: most of them are focused on the conception that creativity deals with generating ideas that are both novel and valuable (Boden 1999). More particularly, creativity has been identified as a meta-level ability, which leads to reflect on and modify our own frameworks and principles (Buchanan 2001). According to that, scientific enterprise is considered to be a typical creative activity since it presupposes and involves both novelty and relevance in the generation of new knowledge.

In the endeavors to grasp the essence of creativity and to investigate its automatic reproducibility, the possibilities of programs and machines that can be called creative have been widely examined (Artificial Intelligence 1997). This fertile research area, called machine creativity, has shown to be an efficient mean also to understand creativity: the implementation level offers the opportunity to test the ideas proposed at the theoretical level. Machine creativity has led, from the one side, to the design of computational supports for creative people and, from the other side, to the conception of computational models of creative processes.

In this paper we afford the theme of creativity in scientific discovery by concentrating on the role of a class of artificial intelligence machines, called *scientific social agencies*. A scientific social agency is a cooperative multiagent system that is able both to *support* creative people, as scientists, in their activities and to *represent* models, namely the creative products resulting from scientific efforts. We analyze two paradigmatic current

scientific projects that show the role of information machines in scientific discovery and the sociality of the scientific discovery activity, thus extending the philosophy of creativity toward that of *social* creativity. The analysis of these two examples of modern scientific practice evidences a list of requirements that can be satisfied by the adoption of a scientific social agency.

This paper is organized as follows. In the next section, we present the concept of scientific social agency. In Sections 3 and 4, we illustrate the two examples from which the desired characteristics of scientific social agency emerge: the Human Genome Project and the Screensaver Lifesaver Project, respectively. The desiderata for scientific social agency are summarized in Section 5, where they are also discussed with respect to the present state of the art. Section 6 proposes a flexible and powerful way to implement scientific social agency in order to fulfill the requirements of Section 5. Finally, Section 7 concludes the paper.

2 Scientific Social Agency

In this section we present the agency as a particular kind of multiagent system and its role within scientific discovery. *Multiagent systems* (Weiss 1999) are systems of distributed artificial intelligence that are composed of several entities, called *agents*, which are spatially distributed and interacting together. Since there is not a global agreed definition of what an agent is (Franklin and Graesser 1997), for our purposes we consider *information machines*, both physical (computer or robot) and logical (software programs), as agents. Agents present the properties of autonomy, social ability, reactivity, and pro-activeness (Wooldridge 1995).

Moreover, agents perform inferential activities (i.e., logical reasoning, see Russell and Norvig (1995)).

When these agents are designed to cooperate, it is profitable to consider the whole multiagent system as a unitary machine called *agency* (see Minsky (1985) for the origin of the name 'agency'). An agency is thus a *cooperation machine*, although the composing agents can have very complex natures (Amigoni et al. 1999b). Each agent of an agency can perform some high-level functions, such as planning a sequence of tasks. Moreover, each agent has the abilities to cooperate with other agents when it is the case. In fact, the agents in an agency interact together to offer or receive collaboration in order to achieve a global goal. The necessary infrastructure for both cooperation and coordination among agents is a communication network that connects the agents together. This communication network can be implemented, for instance, as a shared memory area or as an Intranet.

In order to build an agency, several problems have to be addressed and solved. The first problem is the development of a cooperative organization among agents. Other problems regard how tasks are assigned to agents and how knowledge is exchanged and shared. It is usually possible to develop flexible architectures for agencies, to allow agents to be easily added and removed, promoting a modification of the agency composition to better adapt to the specific task at hand (more details are given in Section 6). We note that an agency can be structured in several nested levels, since it can be composed of subagencies, each one of them acts as a single agent.

Among the applications that can be addressed by agencies, one of the most promising relates to the field of scientific discovery. Usually, in this context, artificial intelligence programs and devices can have a wide range of roles: basically for all of them the purpose is to emulate some human intellectual activities performed during scientific discovery (such as hypothesis construction, theory revision, law induction, and theory formation), promoting what has been called *computer-supported scientific discovery* (de Jong and Rip 1997). The theme of scientific discovery processes has also traditionally represented an area of interest for the philosophy of science, which has constantly tried to explain the mechanisms and the processes presupposed by scientific activity in order to account for the development of scientific knowledge and creativity.

Currently, as exemplified in the following sections, the scientific research can be seen as characterized by two main features:

- (c) the increasing role of information machines as supporting the scientists' activities (computational property);

- (s) the collective and social nature of the scientific enterprise, which is no more carried on by a single scientist in isolation, but by many scientists in collaboration (social property, see Turchetti et al. (2002)).

According to this scenario, an agency can play a very interesting role within creative scientific discovery: as a *scientific social agency* it is a machine that effectively adapts itself and responds to the two features of the contemporary scientific research. More precisely, a scientific social agency is a cooperative multiagent system composed of information machines, the agents. We refer to the agents of a scientific social agency as *man-machine poles* (Amigoni et al. 1999a) to express the idea that men can perform some of their intellectual activities by means of these information machines. Therefore, being the man-machine poles the information machines that support the scientists activities, scientific social agency reflects the feature (c) of current scientific practice. A man-machine pole can be an agency itself, according to what previously said about the nested levels of the agency architecture. Each agent of a scientific social agency is devoted to a particular specialized task in the scientific discovery process. The global performance of a scientific social agency is obtained by the cooperation of the agents that are uniformly integrated. In this way, scientific social agency emphasizes the social nature of scientific enterprise, according to the feature (s) above. Sociality is enlightened by the cooperation of agents that reflects the more general cooperation of the men who are supported by the agents (their man-machine poles).

In the following, we will consider two concrete examples that are representative of the current scientific practice: the Human Genome Project and the Screenshot Lifesaver Project. Since both the fundamental role of information machines and the social character of scientific discovery – the two key marks of current scientific practice – clearly emerge in these paradigmatic examples, we will base on them the discussion of the role of scientific social agency within the process of scientific discovery. In order to better address the description of the complexity of the current scientific practice, in the first example we will emphasize the *vertical* complexity, namely the completely different natures of the several processes involved in the Human Genome Project. In the second example we will emphasize the *horizontal* complexity, namely the large number of parallel computational activities.

3 The Human Genome Project

In this section we illustrate the central features and the main steps of the Human Genome Project (HGP), officially initiated in the United States in 1990 (The Human Genome 2002; Venter et al. 2001). The aim of the project is to decode the DNA (more precisely, the

complete nucleotide sequence) that constitutes the human genome, in order to better understand the human evolution, the causes of diseases, and the interplay between environment and heredity in defining human condition. This scientific effort clearly illustrates the two features (c) and (s) of the contemporary scientific practice. Indeed, the success of this project strongly depends on the availability of various information machines, in particular those implementing computational methods for sequencing the human genome. At the same time, it is a colossal project that involves several different collaborating laboratories and research centers, both public and private.

According to this scenario, a double role for scientific social agency may be envisaged. It can *support* scientists in their activities and it can *represent* the obtained scientific results. In the first role, scientific social agency, according to its nature of concrete, flexible, and powerful machine, represents a practical support for scientists during the process of scientific discovery and is called *assistant (scientific social) agency*. Usually scientists utilize and exploit a number of instruments and tools in carrying on their work. Among these, information machines are in a prominent position since a larger and larger number of not only practical, but also intellectual, activities can be delegated to them both for necessity (e.g., huge quantity of data) and for convenience (e.g., speed increasing). The advent of the first automated DNA sequencers in the HGP is a clear example of this situation. These sequencers have considerably improved the human speed in the sequencing process. In the first years of the project, when the DNA sequencers were not available, a researcher was able to isolate and read 10,000-20,000 bases (the building blocks of our DNA) per day. Nowadays, an automated sequencer is able to process 10 millions bases per day. This is just one of the several examples showing the role of information machines as supports for scientists in their research, promoting the employment of wide and complex computer-supported scientific environments.

An assistant agency can provide a particularly powerful computer-supported discovery environment, where the flexibility of the agency machine can be fully exploited. Besides being a collection of information machines supporting scientists, an assistant agency is conceived as a cooperation machine that offers a valid support for the social and distributed nature of the contemporary scientific research. Let us show this point by referring again to the HGP example. The HGP has always been carried on by a wide distributed net of people organized in research centers: at the beginning (in 1990) only public laboratories, coordinated and financed by the United States National Institute of Health and Department of Energy, were involved in the project. They were relatively independent units that communicated and exchanged relevant information.

Then, the scenario has become even more articulated with the advent of the private venture and, in particular, of Celera Genomics which, starting from 1998, has introduced a burning competition for the completion of the sequencing and mapping of the human DNA 3 billions bases. It is worth noting that, in this example, the distributed character of this scientific enterprise can be found not only along the horizontal dimension, but also along the vertical one. In fact, in the research centers involved in the project, a number of experts in different disciplines, such as molecular biologists, geneticists, and computer scientists communicate and work together to integrate their respective results: This social and collective nature of scientific discovery (both horizontally, among research centers, and vertically, among specialized experts in the same research center) could be efficiently supported by an assistant agency that can exploit its cooperation mechanism.

We consider now the second role of a scientific social agency in which it offers a way for describing the results of the scientific discovery process and is called *representational (scientific social) agency*. More specifically, a representational agency describes, in a concrete way, the set of models resulting from a scientific effort. These models are embedded in the agents of the representational agency, providing a descriptive (when the models are simply stored in the agents) or a more powerful operational (when the models results from the agents activity) representation of the scientific knowledge. The representational agency addresses the two features (c) and (s) of modern scientific discovery, as the HGP example demonstrates. With respect to feature (c), information machines are necessary to store the huge quantity of complex data relative to the genome. A million of bases is equivalent to about 1MB of memory on a computer: since the human genome includes 3 billions bases, around 3GB of memory are necessary in order to contain it, without considering all the notes and the comments which are essential to complete the information describing each gene. It is clearly impossible to manage this enormous amount of sequences information by hand. Moreover, the aim of the HGP is not limited only to identify the 80,000-100,000 human genes and map their positions on the chromosome, but also to determine the role of each protein of DNA in the organism. This can be done only by several cooperating specialized research centers (recall feature (s) of current scientific practice) that call for a representational agency to organize both the collaboration among different contributors and the coordination of their contributions.

4 The Screensaver Lifesaver Project

The two properties we aim to enlighten as the central features of the contemporary scientific enterprise,

namely the computational (c) and the social (s) aspects, are pointed out also by the Screensaver Lifesaver Project (2002). The basic idea of this interesting scientific effort is to accelerate the research for new cancer drugs by means of a kind of distributed software, which enables the spare time of computers to be used to screen molecules for potential anti-cancer activities.

The success of the project, and even its existence, depends on the enormous computational power provided by the parallel work of thousands of computers around the world. They are able to contemporaneously process such a volume of information that is impossible to process by just one computer, even if sophisticated. The idea is that anyone with access to a personal computer could potentially help by donating the "screen-saver time" of his or her computer leading to the creation of a virtual super-computer. Hence, the computational aspect strongly depends on the social one: not only several institutions and scientists are collaborating in order to achieve the goal, but everyone interested in the project can offer his or her collaboration. The project will be the more successful the more people will subscribe it and will make their computers available. This project, due to its peculiar features, can let emerge the usefulness of scientific social agency both in its role of support for scientists - as assistant agency - and in its role of description of the scientific results - as representational agency.

Let us consider in more details the Screensaver Lifesaver Project in order to better explain the concept of assistant agency. The collaboration between the Oxford University Center for Computational Drug Discovery and an American technology company, called United Devices, has led to the creation of the Virtual Center for Computational Drug Discovery, focused in particular to find efficient drugs for cancer. The current anti-cancer therapies are concentrated on proteins supposed to be the target of a cancer therapy. The main purpose of the research is to find molecules that, firstly, inhibit the enzymes which stimulate the blood flow to tumors and, secondly, work against the proteins responsible for cell growth and cell damage. In order to determine potential anti-cancer molecules to be developed as drugs, it is necessary an enormous screening activity.

The difficulty basically lies in the high number of molecules to be screened: there are 2.5 million starting molecules resulted from a preprocessing activity that eliminated the molecules that have not drug-like physical properties (such as solubility, reactivity, and easy metabolization). Moreover, each of these molecules is suitable to generate 100 derivatives made by small changes. Thus, the estimated number of molecules to review for this project is around 250 million for each protein. So far, 12 target proteins (possible responsible of the cancer growth) have been identified: this means

that, totally, about 3 billions of molecules need to be screened. These molecules include not only the commercially available ones, but also many others originated from approximately 20 Combinatorial Chemistry libraries (Murray and Cato 1999). For this purpose, several organizations donated their collections of chemical data and catalogues of molecules to contribute to this project, thus envisaging another perspective on the sociality aspect involved in the project.

Analyzing this quantity of data clearly requires an enormous amount of computational power: traditional information machines are not enough and even supercomputers are limited in supporting scientists in order to evaluate potential anti-cancer molecules. The solution adopted in the Screensaver Lifesaver Project is to exploit the unused computational power of the largest possible number of computers: the necessity of computational power depends on the extended social character of the project and vice versa.

Let us consider how this solution works: at the beginning, each subscribing computer receives an initial package of 100 molecules over the Internet, together with a drug design software application called Think and a model of a target protein known to be involved in causing cancer. Think evaluates the molecules for cancer-fighting potential by creating three-dimensional models of them and testing their interaction with the target protein. This software program, once installed, runs unobtrusively in the background and works on small workloads. This virtual screening of the molecules consists of the evaluation of the many possible shapes, or conformations, the molecules might adopt when interacting with a protein (see Allen et al. (2001) for a similar approach). When a successful conformation dock happens, namely when the molecule triggers an interaction with the protein, this is registered as a hit and sent back to a central server for further investigations. All the hits are recorded, ranked as to strength, and filed for the next stage of the project which is exclusively performed by specialized scientists.

This distributed computer network is not just a traditional computer-supported scientific environment, where scientists are supported in some of their intellectual activities by sophisticated information machines (that perform an automatic screening, as opposite to a manual screening). It is rather a novel way of doing research, which basically tries to exploit the computational resources of common people who are not directly involved in the research. This clearly changes the way medical research, intended as an instance of scientific discovery, is performed. The distributed computing and its coordination represent therefore a primitive instance of an assistant agency, whose agents are the Internet-connected computers processing the molecules, which is suitable, given its architecture, to respond to the requirements settled down by this scien-

tific effort. The analysis of how the public contribution of computing power is managed offers new insights over the role of assistant agency. The P2P (Peer-to-Peer) applications (developed by Intel, which is sponsoring the project) share resources such as hard drives and processing power among the connected computers to significantly increase the computing capabilities. These applications allow the parallel work of millions of individual computers (exactly 1,430,431 on March 5th, 2002) acting simultaneously on different molecules. In this sense, computer owners have the opportunity to use their personal computing resources to perform scientific research. Thus, we may individuate two levels of sociality within this scientific project: the social and distributed interaction between scientists, research centers, public and private laboratories, and technological companies, on the one hand; and the social and distributed interaction between scientists and the rest of the world, intended as the common people who donate the unused power of their machines in order to contribute to the research, on the other hand. The first level of sociality is similar to that of the HGP we have described in the previous section. The second level of sociality characterizes particularly the Screensaver Lifesaver Project and results in the cooperation of computational chemistry, computers, specialized software, organizations, and individuals.

Besides the above depicted primitive assistant agency, scientific social agency can, also in this example, represent the descriptions of the scientific results, acting as a representational agency. The representational aspect, characterized by both property (c) and property (s), involves the results returned by the devices members of the project, which are a first product of the scientific process. In order to record, rank, and file these hits (namely, the molecules that showed an interesting interaction with the target protein) a big computational power is needed. At the moment, a cluster of computers is used to store the results from the screened molecules and to evaluate the molecules that could be developed as drugs. In this context, the Virtual Center for Computational Drug Discovery is the coordinator that collects and manages the various and heterogeneous results coming from the thousands computers spread around the world. It is clear that the results management could be successfully enhanced by employing a representational agency with high-level information fusion capabilities.

5 Desiderata for Scientific Social Agency

The analysis of the two previous examples has evidenced some desiderata for the design and the development of a scientific social agency. We deem that the most important ones are the following.

- The agents must share a unique common language to ensure interoperability among them.
- Efficient mechanisms for the negotiation and the delegation of tasks among the agents must be identified. For example, the agents of the Screensaver Lifesaver Project could benefit from a dynamic efficient redistribution of the computational burden among them, according to their current workloads.
- A global cooperation framework to proactively answer to the needs of the scientists must be developed. For example, the automatic retrieving of the papers and the documents referring to the subjects of the emails received by the users.
- The scaleable reconfiguration of the scientific social agency to allow the easy insertion of new agents (namely, new man-machine poles) and the easy elimination of old agents that are no more useful. For example, it is desirable that the agents supporting the work of a geneticist who joins the HGP could be easily inserted in the scientific social agency that supports the project.

Despite there are several implemented agencies that address different applications, the agency technologies are not yet fully developed to be commonly employed. In particular, the application of an agency as an assistant agency has not been yet tested in real world scientific environments. The currently used computer networks with compatible communication protocols can be considered only as embryonic assistant agencies. As already said, to obtain a “real” assistant agency, high-level cooperative functions (that are still performed by humans) have to be developed: cooperative information retrieval from various sources, cooperative data mining from different databases, information integration based on ontologies, distributed scheduling of the activities, and so on. The research in all these fields is very lively and promising; this supports our opinion that the assistant agency is the next-to-come general and powerful computer-supported environment to strengthen the creativity of scientists in very complex scenarios.

Similarly, although the adoption of a representational agency could offer an improvement in managing the different models produced by the scientific creativity process and in describing the final result, a “real” representational agency is not available. However, it is easy to envisage the advantages it can provide, with its high-level cooperation functions like information fusion, over the present simply-communicating information machines that only store data.

Besides the discussed requirements, we deem that the two different roles of a scientific social agency, namely assistant agency and representational agency, must be

mutually integrated in order to consider the agency as a powerful and flexible machine for scientific creativity. This integration can be promoted at the light of a conceptually very relevant property called *circularity*. Circularity is related to the possibility of implementing both the assistant agency and the representational agency in a unique physical agency which is able to perform both roles. In this way, the representation of new results, provided by the representational agency, and the discovery environment, based on the assistant agency, can mutually improve each other. Some results of the scientific enterprise, represented by the agents of a representational agency, can be physically inserted in an assistant agency. Therefore, this new enhanced machine supports the production of new results that, in turn, are employed to further empower the tool in an endless evolutionary process.

To better illustrate the circularity property we are advocating, let us consider again the HGP scenario. The results of the HGP are stored in a collection of databases that record, organize, and interpret the flood of data emerging from sequencing projects worldwide. GenBank is, for instance, a genetic sequence database that includes an annotated collection of all publicly available DNA sequences. It collects approximately 15,850,000,000 bases in 14,976,000 sequence records (December 2001) and receives every day new data that are checked and inserted in the archive (that is publicly accessible via Web at The Human Genome (2002)). These connected databases can be considered as an embryonic representational agency since high-level cooperative functions are not yet developed, although a unique format for exchanging data (that is, the base of cooperation) already exists. The description of the scientific results expressed in this simple representational agency allows, according to the circularity property, the development of an improved assistant agency. The agents composing this assistant agency, the DNA sequencers for example, rely on the previous results embedded in the agents of the representational agency to prevent the repetition of the work already done by others.

The same circularity property can enhance also the Screensaver Lifesaver Project. The returned results, namely the hits of the successful interaction between the molecules and the target protein, are recorded, ranked, and filed by the cluster of the Virtual Center for Computational Drug Discovery, which can be seen as an embryonic representational agency. These results are then utilized by chemists and molecular biologists, according to circularity property, in order to develop better assistant agencies, both at the level of the analysis of the interaction between molecules and proteins and at the level of the design and the development of anti-cancer drugs.

We summarize some of the stated desiderata, and in particular the mutual interaction between the two roles of scientific social agency, in the diagram of Figure 1.

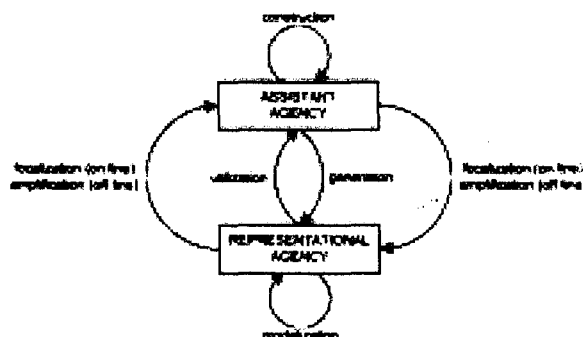


Figure 1 – The graphic representation of the flexibility requirements on scientific social agency

The diagram shows the assistant agency that, as already discussed, supports scientists in the *generation* of creative results and the representational agency that allows the *utilization* of the creative results. The circularity property is illustrated by the two arrows from assistant agency to representational agency and vice versa. The two loops on assistant agency and representational agency illustrate the dynamicity that is desirable in the management of their agents (see the last point of the desiderata list). The circularity property is further discussed in the following section.

6 Dynamic Agency

Given the desiderata for scientific social agency, as shown in Figure 1, in this section we propose a way in which a scientific social agency could be developed in order to fulfill these requirements.

The dynamicity and the flexibility in the management of the agents of a scientific social agency could be reached by adopting the *dynamic agency* methodology we have developed, as described in Amigoni and Somalvico (1998), to build flexible multiagent systems. The dynamic agency approach (that has been proved to be successful for robotic applications, see Amigoni and Somalvico 2002) relies on a specific architectural structure of each agent as divided into two parts. The first part, called *op semiagent*, is composed of the hardware and of the basic software components of the computer (or of the robot). These components are devoted to *operation*: the op semiagent exhibits the abilities to operate in the environment. For example, the hardware components include sensors, actuators, processing units, and communication devices; the software components include control systems for sensors and actuators, operating systems of processing units, pro-

grams for managing the communication protocols, and so on. The second part, called *co semiagent*, is composed of high-level software modules. These modules are devoted to *cooperation*: the co semiagents are oriented to integrate the op semiagents in a uniform and coherent cooperation framework. For example, they provide functions for the negotiation and the division of tasks, for high-level knowledge exchanging, and so on. Hence, in the dynamic agency approach, each agent of a multiagent system is composed of the op semiagent and of the co semiagent.

The original and powerful way we propose to implement the dynamic agency architecture is to realize the software modules of the co semiagents exploiting the modern technique of mobile code systems (Fuggetta 1998; Picco 2001). These systems allow to build *execution units* (namely, software processes) that can migrate in a network from one host to another, and resume their execution from the point they interrupted. In our methodology, the software modules of the co semiagents are spread on the op semiagents by a unique execution unit that replicates and evolves on each one of them. In this scenario, the op semiagents are the hosts (in a network) on which the execution unit runs. To stress the mobility feature, we call *Mobile Intelligent Agent (MIA)*, the execution unit that constructs the co semiagents (see Figure 2). By contrast, the op semiagents have to show the ability to host and execute the MIA. Hence, in the dynamic agency methodology, we divide the adapting of a computer (or a robot) to be integrated in a multiagent system, obtained by allowing the op semiagent to host the MIA, from the building of the cooperation mechanism among agents, obtained by spreading the co semiagents. We outline that the construction of the co semiagents as the evolution of a replicated MIA is one of the distinctive features of the dynamic agency methodology that enables the easy management of the co semiagents.

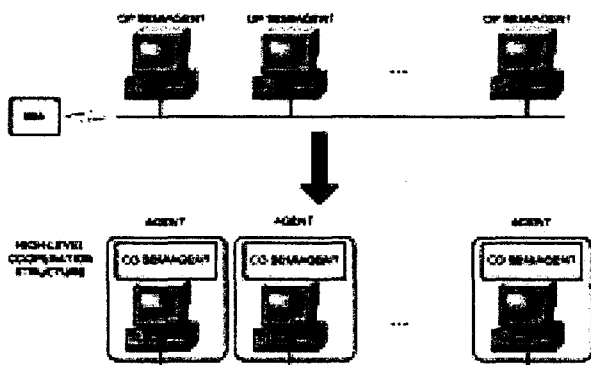


Figure 2 – The MIA is sent on the communication network connecting the op semiagents (top) in order to spread uniform co semiagents that set up an high-level cooperation structure (bottom)

The definition of the structure of the MIA and of the hosting abilities of the op semiagents are the tasks the designer of a multiagent system has to undertake. The activities of the MIA are designed to install the co semiagents on the op semiagents and to control them in order to cooperatively reach a global goal. In this way, the operative functions provided by the op semiagents are exploited by the high-level cooperation structure set up by the co semiagents that exchange knowledge, coordinate the activities, and divide and assign goals.

The advantages of adopting mobile code systems for implementing the dynamic agency methodology are summarized as follows.

- Independence of the designers of the op semiagents (computers or robots) from the designer of the whole multiagent system. Since the composing op semiagents are taken 'as they are', the multiagent system designer has the possibility to integrate in the system several different op semiagents that enrich the multiagent system, allowing it to tackle a broad problem spectrum.
- Easy reuse of the existing op semiagents for different purposes. In dynamic agency approach, existing computers and robots are considered as parts of op semiagents on which different co semiagents can be installed at different times in order to build various multiagent systems. These exploit the operative functions of the computers (or robots) for addressing different applications.
- Automatic reconfiguration of the multiagent system. The idea is that the process of installing the co semiagents on the op semiagents can be performed dynamically also during the operation of the system. In this way, a new agent can be dynamically integrated in the multiagent system allowing, therefore, a dynamic reshaping of the system that can improve its effectiveness in tackling complex problems. In a similar way, an agent can be dynamically excluded from the system.
- Economic advantages. Since the computers (or robots) are viewed as existing elements, offering operative functions and possibly developed from third parties, in the future we will face a situation somehow similar to that of object oriented programming (Martin and Odell 1998), in which op semiagents may play the role of library classes, co semiagents may play the role of user-developed classes, and complete agents may play the role of programs. If the idea of dividing the operative part from the cooperative part is pushed further on, it is possible to envisage a scenario in which the op semiagents are developed in large quantities at low-cost and are employed

to build increasingly complex multiagent systems.

In conclusion, we stress that the dynamic agency approach presented in this section is a good candidate for developing a scientific social agency that meets the requirements discussed in Section 5 and, in particular, those related to the easy management of the agents and to the circularity property. Referring again to Figure 1, in the case of the assistant agency, the dynamic agency approach enables the definition of a learning evolutionary process, called *construction*, which constructs the most appropriate architecture of the assistant agency. In fact, the assistant agency could include a redundant set of agents, which could not be useful for the specific creative process currently undertaken. Hence, during the ongoing scientific process, an adaptive selection of useful agents of the assistant agency can be performed to better tune the architecture of the assistant agency. Similarly, in the case of representational agency, the dynamic agency approach enables the definition of a learning evolutionary process, called *modelization*, that ends in the final model of the scientific results composed of the most appropriate agents. In fact, the representational agency could include a redundant set of agents, which could not be useful for the current representation of scientific results. Also in this case, during the ongoing shaping of a representation, an adaptive selection of useful agents of the representational agency can be performed to better refine the global model embedded in the representational agency.

Given these two dynamic evolutionary processes, we can envisage two different ways in which the circularity property could be carried out (see Figure 1). In the first case, called *focalization*, the assistant and the representational agencies enhance each other on line, namely during a scientific creative process. In the second case, called *amplification*, the assistant and the representational agencies improve each other off line, namely at the end of a creative scientific process. In both cases, the circularity property accounts for the bottom-up improvement of the assistant agency, according to the new results described by the representational agency, and for the top-down improvement of the representational agency, according to the activity of the assistant agency.

7 Conclusions

We have presented scientific social agency as an interesting and powerful device to enhance (more and more as agency technologies further develop) scientific discovery considered as a form of creativity. Scientific social agency accounts both for the increasing role of information machines within scientific discovery and for the social character of the scientific enterprise, which we consider as the main features of the contemporary scientific research. Moreover, we have shown

how scientific social agency is able to play two roles: as support for scientists, assistant agency, and as description of scientific results, representational agency, which are integrated by the circularity property. We explicitly note that our approach toward scientific creativity does not provide a model of the whole process of scientific discovery. We do not claim that scientific social agency is a creative machine, namely a machine which performs autonomously some creative activities; it is rather a machine that strengthens human creativity.

In the future we plan to enrich the scientific social agency paradigm to evidence the role of the various levels involved in its architecture and the properties it exhibits when it is composed of homogeneous (as in the Screensaver Lifesaver Project of Section 4) or heterogeneous (as in the HGP example of Section 3) agents. The long-time goal of our efforts is to experimentally demonstrate the usefulness of scientific social agency. To this end, we are currently working on methods and techniques to transform a measurement system in a perceptive agency. This represents one of the basic stones on which a natural science oriented scientific social agency will be developed.

References

- B. C. P. Allen, G. H. Grant, and W. G. Richards. Similarity Calculations Using Two-Dimensional Molecular Representations. *Journal Chem. Inf. Comput. Sci.*, 41:330-337, 2001
- F. Amigoni and M. Somalvico. Dynamic Agencies and Multi-Robot Systems. In *Distributed Autonomous Robotic Systems 3*, T. Lueth, R. Dillmann, P. Dario, and H. Worn (eds.), Springer-Verlag, Berlin Heidelberg, Germany, 1998, pp. 215-224
- F. Amigoni, V. Schiaffonati, and M. Somalvico. Processing and Interaction in Robotics. *Sensors and Actuators A: Physical*, 72(1):16-26, 1999a
- F. Amigoni, M. Somalvico, and D. Zanisi. A Theoretical Framework for the Conception of Agency. *International Journal of Intelligent Systems*, 14(5):449-474, 1999b
- F. Amigoni and M. Somalvico. Application of Mobile Code to Development of Cooperative Multirobot Systems. *Proceedings of the 7th International Conference on Intelligent Autonomous Systems (IAS-7)*, Marina del Rey, CA, USA, March 25-27, 2002
- Artificial Intelligence, Special issue on scientific discovery*, 91, 1997

- M. Boden. Computer Models of Creativity. In *Handbook of Creativity*, R. J. Sternberg (ed.), Cambridge University Press, UK, 1999, pp. 351-372
- B. Buchanan. Creativity at the Metalevel. *AI Magazine*, Fall: 13-28, 2001
- H. de Jong and A. Rip. The computer revolution in science: steps towards the realization of computer-supported discovery environments. *Artificial Intelligence*, 91:225-256, 1997
- S. Franklin and A. Graesser. Is it an agent, or just a program?: a taxonomy for autonomous agents. In *Intelligent Agents III: Agent Theories, Architectures, and Languages*, J. Müller, M. Wooldridge, and N. Jennings (eds.), Springer-Verlag, Berlin Heidelberg, Germany, 1997, pp. 21-35
- A. Fuggetta, G. P. Picco, and G. Vigna. Understanding Code Mobility. *IEEE Transactions on Software Engineering*, 24(5):342-361, 1998
- J. Martin and J. J. Odell. *Object-Oriented Methods: A Foundation (UML Edition)*. Prentice Hall, 1998
- M. Minsky. *The Society of Mind*. Simon & Schuster, New York, USA, 1985
- C. M. Murray and S. J. Cato. Design of Libraries to Explore Receptor Sites. *Journal Chem. Inf. Comput. Sci.*, 39:46-50, 1999
- G. P. Picco. Mobile Agents: An Introduction. *Journal of Microprocessors and Microsystems*, 25(2):65-74, 2001
- S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995
- Screen saver Lifesaver Project.
<http://www.chem.ox.ac.uk/curecancer.html>, last access March 2002
- The Human Genome .
<http://www.ncbi.nlm.nih.gov/genome/guide/human>, last access March 2002
- S. Turchetti, M. Capocci, and E. Gagliasso. Production, Science and Epistemology: An Overview on New Models and Scenarios. In *Model-Based Reasoning: Science, Technology, Values*, L. Magnani and N. J. Nersessian (eds.), Kluwer Academic / Plenum Publisher, 2002
- C. Venter et al. The sequence of the Human Genome. *Science*, February:1304-1351, 2001
- G. Weiss. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, 1999
- M. Wooldridge and N. R. Jennings. Intelligent Agents: Theory and Practice. *Knowledge Engineering Review*, 10(2):115-152, 1995

Supporting Creativity in Software Design

Paulo Gomes, Francisco C. Pereira, Nuno Seco, José L. Ferreira, Carlos Bento

CISUC – Centro de Informática e Sistemas da Universidade de Coimbra. Departamento de Engenharia Informática,

Polo II, Universidade de Coimbra. 3030 Coimbra

{pgomes, camara}@dei.uc.pt, nseco@student.dei.uc.pt, {zeluis, bento}@dei.uc.pt

Abstract

The software design phase is nowadays a development phase requiring more focus from the software engineers. New design methodologies must be developed with the goal of optimising the software design and the resources needed for the development. Software design is still more an art than an engineering task with clearly pre-defined processes and methodologies. Those methodologies that exist are general guidelines, which can be interpreted in several ways according to the context involved. Like architects, software designers frequently use their experience from the development of previous systems to design new ones. Most of the mature engineering fields make the reuse of components a development rule, but, in software engineering, the reuse of components and/or design ideas is not easy, given the conceptual complexity at hand. Thus, intelligent tools capable of supporting software design are needed. These tools must provide not only intelligent search functionalities, so that relevant software pieces are found, but also be able to explore areas of the design search space not usually explored. In this paper we focus in this last issue by presenting a design tool capable of supporting creativity. This tool enables the exploration of regions in the design space not usually used in the generation of designs. Our approach relies on analogical reasoning to do the exploration. We use a general ontology (WordNet) to support this exploration by providing object classification, and an index structure that can be used to perform the search.

1 Introduction

Software design has been acquiring more importance as the complexity level of software increases. This also drives development teams to be more efficient and more creative in their solutions. Software designers must find new design methodologies, trying to optimise development time, processing time, required memory, and other resources. This kind of design is still more an art than an engineering task with clearly pre-defined processes and methodologies. Those methodologies that exist are general guidelines, which can be interpreted in several ways according to the context involved. Like architects, software designers frequently use their experience from the development of previous systems to design new ones. Most of the mature engineering fields make the reuse of components a development rule, but, in software engineering, the reuse of components and/or design ideas is not easy, given the conceptual complexity at hand. Thus, intelligent tools that support the software design task are strongly welcome. These tools must implement common software reuse (Meyer 1987; Prieto-Diaz 1993; Coulange 1997) techniques, but they also have to go further, providing support for more complex reasoning abilities and exploration of new design spaces.

Design can be divided into routine and non-routine design (Gero 1994). We use the term routine design when the designer is *a priori* acquainted with the necessary knowledge. In this way, routine design comprises parametric variations of old designs. Non-routine design involves two sub-classes: (1) innovative

and (2) creative design. In innovative design, a new artefact is generated from a known class of designs, but differs from previous class artefacts in some substantive way. This can occur, for instance, in terms of value(s) assigned to the design variables, which are specific to the new solution. In creative design, the space of possible design alternatives is expanded through the use of new variables. This takes place when the space of potential designs is extensive and is not entirely defined by the knowledge available to the user. In creative design, the process of specifying requirements is also viewed as an important phase (Kolodner and Wills 1993). Creativity is defined as a cognitive process that generates a product that is said to be creative when satisfying certain kinds of properties or attributes (Dasgupta 1994). As far as we are aware, the phenomenon known as Creativity has at least four components (Brown 1989): creative process, creative product, creative person or entity, and creative situation. Within this model, the creative product is a crucial component. Thus an important issue in developing computational models of creativity is the evaluation of the creative product. Research in this domain has come up with several methods for generation of artefacts considered creative (Gero and Maher 1993; Gero 1994; Partridge and Rowe 1994). Some methods identified in creative reasoning are cross-domain transfer of ideas (e.g., by analogy), combination of ideas and exploration and transformation of conceptual spaces. Most of these methods are used in creative design (Gero 1994).

Most of the software reuse tools (Basset 1987; Prieto-Diaz 1991; Katalagarianos and Vassiliou 1995;

Fernández-Chamizo, González-Calero et al. 1996) support only the retrieval of components (e.g. classes, functions or specifications) from repositories. But reusing software involves also adaptation of the retrieved component to the system being developed. This is usually left to the designer, since it is a more complex and demanding task. Analogical reasoning (Gentner 1983; Hall 1989; Holyoak and Thagard 1989; Bhatta and Goel 1997; Bhatta and Goel 1997) is a technique that can be used in the adaptation phase of software reuse (Maiden and Sutcliffe 1992; Jeng and Cheng 1993; Spanoudakis and Constantopoulos 1994; Tessem, Bjornestad et al. 1994). The transfer of ideas and solutions from other domains to the target domain not only provides a way to find solutions, but also gives the opportunity to explore new solutions, sometimes finding creative designs. Not only because they are novel and unexpected, but also because they are more simple and efficient. We believe analogical reasoning can achieve this on its own or with the collaboration of the software designer, providing the designer with ideas and alternatives that help explore the solution space in a more efficient way.

Being able to explore the huge design space for Object-Oriented software is not an easy task. A software design system capable of coping with this exploration task, must be able to handle a huge amount and diversity of domain knowledge. There are two main solutions for this problem: either it is assumed a wide pre-defined categorization (in order to classify each situation), thus organizing the knowledge base in some well-defined and understandable way (e.g. a domain taxonomy), or some kind of open-ended characterization is allowed, subject to an underlying common ontology (e.g. a common sense ontology). While the former demands a huge knowledge engineering effort with the obvious limitation of never guaranteeing completeness, nevertheless facilitating the system's organization and performance, the latter needs less effort in constructing the base ontology but a much greater effort in structuring and constructing the knowledge base in order to achieve some degree of robustness and organization. In terms of creativity potential, we believe diversity will bring a very positive contribution, although a lot harder to control. At this point, we must say that half of the success is guaranteed by the competence of the underlying ontology base. The well-known project WordNet (Miller, Beckwith et al. 1990), by its generic scope, organization and continuous development, seems to us a very good choice for such a task.

We are developing a CASE (Computer Aided Software Engineering) tool, ReBuilder, which applies analogical reasoning to the reuse and design of object-oriented software. Our goals are the development of an intelligent tool for supporting software design, and to motivate the generation of creative solutions in collaboration with the designer. This can be achieved

through the suggestion of new solutions by ReBuilder or by stimulation of the designer's divergent thought (Guilford 1968). To provide the user with an intuitive and commonly used language, we represent software designs in Unified Modelling Language (UML) (Rumbaugh, Jacobson et al. 1998). This is a graphical language used to describe and document object oriented software, and is a standard for most of the software development companies. Using UML solves the man-machine communication problem, which could compromise the use of ReBuilder platform.

In the next section we present ReBuilder architecture, and describe its modules. Then we focus in the analogical reasoning module, explaining it in detail. We also describe how WordNet is used with the analogical engine. Then we present an example of the analogical reasoning. Finally we present some conclusions and future work on ReBuilder.

2 REBUILDER

UML comprises several types of diagrams capable of describing numerous aspects of a software system. These range from requirement analysis using Use Cases, structural information using Class Diagrams, to behavioural knowledge specified using State Machines. In ReBuilder, only UML class diagrams are used for reasoning, though in future work other diagram types can be also used. An example of a class diagram is presented in Figure 1.

ReBuilder comprises four main modules: UML editor, Knowledge Base Management, Knowledge Base (KB), and CBR engine. The UML editor is the front-end of ReBuilder and the environment where the software designer develops the software design. The Knowledge Base Management module is used by the administrator to manage the KB, keeping it consistent and updated. The KB comprises four different parts: the case library, which stores the cases of previous software designs; an index memory that is used for efficient case retrieval; the data type taxonomy, which is an ontology of the data types used by the system; and WordNet, which is a general purpose ontology. From WordNet, ReBuilder uses only the hypernym/hyponym relations (also known as *is-a* relations), and the holonym/meronym relations (which are *part-of*, *member-of*, and *substance-of*).

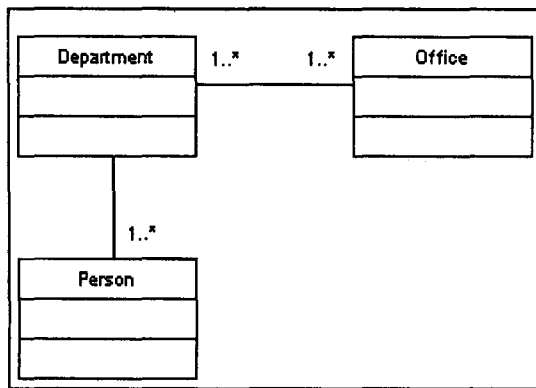


Figure 1 - ReBuilder's architecture.

The CBR engine comprises five sub modules: retrieval, analogy, adaptation, verification and learning. Retrieval is used to suggest past designs, which are selected by the system based on the similarity with the current problem. But the designer can go a step further, and ask the system to complete the current design she/he is working on. For this, ReBuilder offers two different ways of doing it. The first one is using the adaptation module, where the system uses the most similar cases to the working design and tries to complete it with parts of the retrieved designs. But it can also go further and suggest alternatives using the analogy module, which establishes analogical mappings between the working design and one from the case library. After the mapping is established, the system transfers knowledge from the selected design to the current design. The verification sub module performs a check on the working design, checking the design for consistency, completeness, and correctness. For this, ReBuilder uses several object oriented verification rules, and domain knowledge provided by WordNet relations. Finally, the learning module can learn new knowledge from two sources: user interaction and adaptation or analogy generated knowledge.

3 Analogy in REBUILDER

Designing an object model using UML involves two main steps. The first one is creating the class diagram with the classes and interfaces needed to implement the software system. This phase is called the conceptual design of the system's structure and involves only the main entities of the system represented as classes. In the second phase, the designer needs to specify the class attributes and methods. This step requires from the designer a preview of which attributes and methods are necessary for the system implementation, which is not always easy. Analogical reasoning can be used to assist the designer in both these steps.

3.1 Analogy for Conceptual Design and Component Specification

The analogical engine helps the designer by providing new objects¹, attributes or methods for the query design diagram. The new suggested knowledge is based on (partial) mappings between the working class diagram (the query diagram) and class diagrams stored in the case library. The process comprises four steps: select cases from the KB; map the working design to these cases, yielding a mapping for each one; transfer knowledge from the cases to the working design; and finally select the best k new designs (k is chosen by the designer).

In the first phase, the analogical engine searches and identifies a set of candidate cases from the case library. In this step, we use the retrieval sub-module to select the best j cases (j is also defined by the designer). These cases are seed cases for the analogy module. Retrieval is based on the similarity between diagrams, which has two main aspects: conceptual similarity and structural similarity. The conceptual similarity of two objects is based on its WordNet categorization, and the objects' attributes and methods. Structural similarity is computed based on the objects' relations.

The second phase establishes a mapping between each base case and the query diagram. To accomplish this, ReBuilder uses two alternative structure-matching algorithms: one based on relations between objects, and the other based on objects. The first algorithm uses the relations from the query diagram to guide the mapping (see below).

- Relations \leftarrow Get the best relation from the query diagram, based on the independence measure² of the relation.
- MappingList \leftarrow Empty list
- While Relations is not empty Do
 - PRelation \leftarrow Get best relation from Relations, based on the independence measure of the relation
 - CRelation \leftarrow Get best matching relation from the base case relations that are matching candidates (structural constraints must be met)
 - Mapping \leftarrow Get the mapping between objects, based on the mapping between PRelation and CRelation
 - Remove PRelation from Relations

¹ UML objects are: packages, classes or interfaces. Packages are UML object containers, classes possess attributes and methods, and interfaces possess only methods.

² The independence measure is a heuristic based on the UML semantic to define for each diagram object which is its degree of independence regarding the other diagram objects.

- Add Mapping to MappingList
- Add to Relations all the same type³ relations adjacent to PRelations, which are not already mapped (if PRelation connects A and B then the adjacent relations of PRelations are the relations in which A or B are part of, excluding PRelation).
- Return MappingList

The alternative to the relation-based mapping algorithm is an algorithm based on objects, which is an algorithm where the starting points for the mapping are the objects instead of the relations. The algorithm is:

- Objects ← Get object from the query diagram which has the highest independence measure
- MappingList ← Empty list
- While Objects is not empty Do
 - PObject ← Get best object from Objects, based on the object's independence measure
 - CObject ← Get best matching object from the base case objects that are matching candidates (structural constraints must be met)
 - Mapping ← Get the mapping between PObject and CObject
 - Remove PObject from Objects
 - Add Mapping to MappingList
 - Add to Objects all the objects adjacent to PObject, which are not already mapped (and adjacent object B to an object A, are all the objects that have a relation with A).
- Return MappingList

The third phase consists on completing the working design using the selected mappings. In this step, a new design is created for each mapping of the previous phase. It is generated by first making structural transfer of knowledge between the query diagram and the base case. This consists on getting new relations and objects from the base case to the query diagram, according to the structural constraints of the mappings that were established. After this, for each mapped object, a matching between attributes and methods is performed. Each unmatched attribute or method in the base case object is copied to the query object, ending the knowledge transfer.

The last step comprises the ranking of the alternative generated designs. In this task we use four alternative selection criteria: number of interconnected mapped objects, number of mapped objects, sum of the independence measure of the mapped problem objects, or number of objects in the new design.

³ In UML there are four relation types: associations, generalizations, dependencies, and realizations. Only relations with the same type can be mapped.

3.2 Using WordNet as Background Knowledge

In the mapping algorithms, knowing which objects can be mapped is a crucial factor. To accomplish this task, ReBuilder uses the WordNet ontology. In this network, there are relations between synsets (a synset is a set of synonym words expressing the same concept), which reflect a hierarchy of categorization that is used to compute a conceptual distance between two objects. These are the is-a relations (e.g., a human is-a mammal, which is-a animal). Because each object corresponds ultimately to a specific synset (context synset), and since it is normal to have several distinct candidates, we use the object's name to get the right synset. We also use the context synsets of the objects in the same context (class diagram), to choose the correct synset (this process is named disambiguation).

ReBuilder ranks candidate objects using three measures. To illustrate this, consider a target object with context synset A, and a candidate object with context synset B. Suppose that MSCA is the Most Specific Common Abstraction synset between A and B.

- Taxonomical distance similarity – measures the similarity between synset A and synset B, passing by MSCA, based on the taxonomical distance. The higher the value, the more similar are the synsets, and closer they are.

$$1 - \frac{D(A, MSCA) + D(B, MSCA)}{2 \times \text{MaximumDepth}}$$

Where D(A, MSCA) is the distance measured in is-a links between A and MSCA, and MaximumDepth is the is-a taxonomy maximum depth.

- Equilibrium measure – relates to the equilibrium degree of the tree composed by all synsets between A and MSCA, and all synsets between B and MSCA. The closer the value is to one, the more balanced are the distances between A-MSCA, and B-MSCA.

$$1 - \frac{|D(A, MSCA) - D(B, MSCA)|}{\sqrt{D(A, MSCA)^2 + D(B, MSCA)^2}}$$

- Absolute depth measure – relates to the taxonomical depth of MSCA. The closer to one, the deeper MSCA is located.

$$\frac{\text{Depth}(MSCA)}{\text{MaximumDepth}}$$

Where Depth(MSCA) is the minimal distance from MSCA to a taxonomical root synset. The analogical algorithm then uses these three measures in a weighted sum to select the best mapping object from a list of candidates.

ReBuilder uses the object's name to determine the context synset associated to each object. This is a complex task, requiring extraction of all the words from the object's name, and disambiguation of these word

meanings. The first thing to do is to get all the words from the object's name. Remember that, most of the times, an object's name is a composition of two or more words (e.g. *ClientRecord*, *SchoolDepartment*), and these must be identified and separated⁴. After a list of words has been defined, then the algorithm gets the synset list for each word and joins all the synsets in one list (candidate synsets). Then the system computes the sum of the semantic distances between each candidate synset and each context synset of the objects in the same class diagram. The candidate synset with the smaller semantic distance is the one chosen for the object's context synset. These objects comprise the specific context in which the object is located. Word decomposition is performed using a set of detachment rules, which provide a morphological analysis of each word. The semantic distance used in the disambiguation is defined as the number of WordNet relations between the two synsets. The relations used in this distance are is-a, part-of, member-of, and substance-of relations.

3.3 An Example

Figure 3 shows a short example of the application of ReBuilder's analogy module. The UML class diagram presented in Figure 1 is used as the query for the analogical engine. The class diagram of Figure 2 was chosen from the UML case repository and is used to complete the query design. The resulting diagram is presented in Figure 3.

In the first phase, the retrieval algorithm selected several diagrams, from which the one with the highest score is presented in Figure 2. Then, the analogy module used the object-based structural mapping algorithm to establish the mapping between objects. The algorithm starts the mapping selecting the query object with the highest independence measure, which is *Department*. Because no mappings take place in this first iteration, all objects from the base diagram are mapping candidates.

Using WordNet to identify the context synsets, *Department* has the same synset as *SchoolDepartment* - both are considered departments. The second candidate is *School*, and then *Teacher* and *Student*. The algorithm selects *SchoolDepartment* for the first mapping (*Department-SchoolDepartment*). After the mapping is done, the *Objects* list gets one element: [(*Person*, {*Student*, *Teacher*})]. This means that *Person* can be mapped to *Student* or to *Teacher*, according to the diagram relations that act as structural constraints.

⁴ We are aware that, in English, the meaning of a compound name (a name composed of two or more names) cannot often be built from its parts. Yet, in the context we are dealing here, this process of disambiguation leads, if not to the exact meaning, at least towards the correct context.

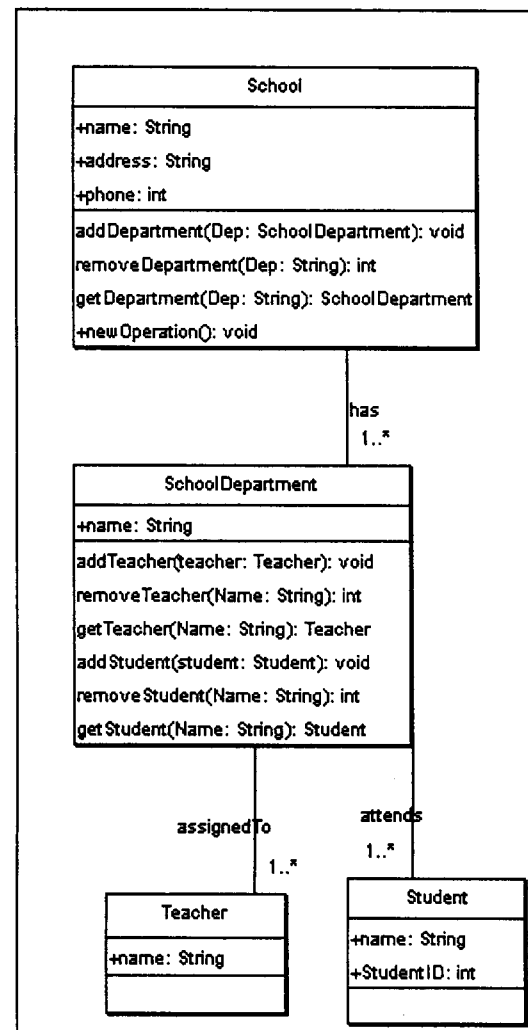


Figure 2 - The source UML class diagram used to complete the class diagram of Figure 1.

In the next iteration, this element is retrieved from the *Objects* list, and the algorithm uses WordNet to select the best candidate. Since the path in WordNet between *Student* and *Person* is 3 arcs long (*Student* is-a *Enrolee* is-a *Scholar* is-a *Person*) and between *Teacher* and *Person* is 4 (*Teacher* is-a *Educator* is-a *Professional* is-a *Adult* is-a *Person*), the selected candidate is *Student* (*Person-Student*). At this point, the algorithm has no more candidate mappings because of the structural constraints of the diagram.

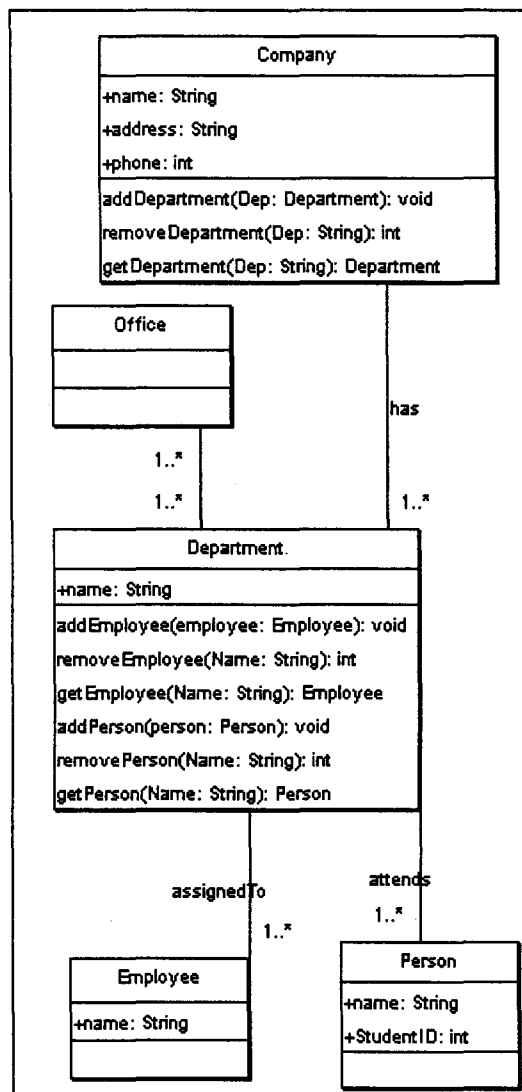


Figure 3 - The result of analogical reasoning between diagrams of Figure 1 and Figure 2.

After the mapping between diagrams, the analogy module transfers the knowledge from the base diagram to the query diagram. First the relations and objects associated with the mapped objects are transferred from the base diagram to the query diagram, always according to the structural constraints. In this case, two new classes were transferred to the query diagram, which had to be renamed by the designer (*School* to *Company*, and *Teacher* to *Employee*). Then, the attributes and methods of the base objects are transferred into the respective query objects. In Figure 3 *Department* gets the methods and attributes from *SchoolDepartment*, and *Person* the attributes of *Student*.

4 Conclusions and future work

In this paper, we presented a CASE tool that uses analogical reasoning to enable the software designer creativity. By providing a design space exploration mechanism along with large broad-spectrum ontology, Re-

Builder can suggest alternative solutions to the software designer. This enables the designer to be guided in the design space exploration, evaluating the designs generated by the system. Using a human-created design language like UML, allows ReBuilder to be a broadband software design tool. Also with the intent of being an easy to use tool, ReBuilder integrates a general ontology like WordNet, so that the designer can be understood by the machine, instead of having to explain himself to it.

One of the main advantages of analogical reasoning is its capability to explore different domains, and to create new ideas from this exploration. Despite this major benefit, analogical reasoning has some limitations, such as the complexity and expensive computational work involved in the process, and also the creation of bizarre designs. Some of these problems can be solved using search-guiding heuristics, so that the search done by the analogical reasoning can be oriented to productive areas of the solution space. Another way to solve this problem is by using a common sense ontology like WordNet to validate the solutions generated by the analogy module. In future work we plan to implement this validation mechanism in ReBuilder using WordNet semantic relations. Future work on ReBuilder will also involve the use of Design Patterns as Case-Based Reasoning adaptation plans for software designs.

5 Acknowledgments

This work was partially supported by POSI - Programa Operacional Sociedade de Informação of Portuguese Fundação para a Ciência e Tecnologia and European Union FEDER, contract POSI/33399/SRI/2000, by program PRAXIS XXI.

6 References

- Basset, P. G. (1987). "Frame-Based Software Engineering." *IEEE Software*(July): 9-16.
- Bhatta, S. and A. Goel (1997). *An Analogical Theory of Creativity in Design*. International Conference on Case-Based Reasoning (ICCBR 97), Providence - Rhode Island, USA, Springer-Verlag.
- Bhatta, S. and A. Goel (1997). *Design Patterns; A Computational Theory of Analogical Design*. International Joint Conference on Artificial Intelligence (IJCAI'97).
- Brown, R. (1989). Creativity: What are We to Measure? *Handbook of Creativity*. J. Glover, R. Ronning and C. Reynolds, Plenum Press.
- Coulange, B. (1997). *Software Reuse*. London, Springer-Verlag.
- Dasgupta, S. (1994). Creativity, Invention and the Computational Metaphor: Prolegomenon to a Case Study. *Artificial Intelligence and Creativity*. T. Dartnall, Kluwer Academic Publishers.

- Fernández-Chamizo, C., P. González-Calero, et al. (1996). *Supporting Object Reuse through Case-Based Reasoning*. Third European Workshop on Case-Based Reasoning (EWCBR'96), Lausanne, Suisse, Springer-Verlag.
- Gentner, D. (1983). "Structure Mapping: A Theoretical Framework for Analogy." *Cognitive Science* 7(2): 155-170.
- Gero, J. (1994). Computational Models of Creative Design Processes. *Artificial Intelligence and Creativity*. T. Dartnall, Kluwer Academic Publishers.
- Gero, J. (1994). Introduction: Creativity and Design. *Artificial Intelligence and Creativity*. T. Dartnall, Kluwer Academic Publishers.
- Gero, J. and M. L. Maher (1993). *Modelling Creativity and Knowledge-Based Creative Design*. Sydney, Lawrence Erlbaum Associates.
- Guilford, J. (1968). *Intelligence, creativity and their educational implications*. San Diego, CA, Robert Knapp.
- Hall, R. P. (1989). "Computational approaches to analogical reasoning; A comparative analysis." *Artificial Intelligence* 39(1): 39-120.
- Holyoak, K. J. and P. Thagard (1989). "Analogical Mapping by Constraint Satisfaction." *Cognitive Science* 13: 295-355.
- Jeng, J.-J. and B. Cheng (1993). *Using Analogy and Formal Methods for Software Reuse*. IEEE 5th International Conference on Tools with AI.
- Katalagarianos, P. and Y. Vassiliou (1995). "On the reuse of software: a case-based approach employing a repository." *Automated Software Engineering* 2: 55-86.
- Kolodner, J. and L. Wills (1993). *Case-Based Creative Design*. AAAI Spring Symposium on AI+Creativity, Stanford, CA, USA.
- Maiden, N. and A. Sutcliffe (1992). "Exploiting Reusable Specifications Through Analogy." *Communications of the ACM* 35(4): 55-64.
- Meyer, B. (1987). "Reusability: The Case for Object-Oriented Design." *IEEE Software* 4(2, March 1987): 50-64.
- Miller, G., R. Beckwith, et al. (1990). "Introduction to WordNet: an on-line lexical database." *International Journal of Lexicography* 3(4): 235 - 244.
- Partridge, D. and J. Rowe (1994). *Computers and Creativity*, Intellect Books.
- Prieto-Diaz, R. (1991). "Implementing Faceted Classification for Software Reuse." *Communications of the ACM*(May).
- Prieto-Diaz, R. (1993). "Status Report: Software Reusability." *IEEE Software*(May).
- Rumbaugh, J., I. Jacobson, et al. (1998). *The Unified Modeling Language Reference Manual*. Reading, MA, Addison-Wesley.
- Spanoudakis, G. and P. Constantopoulos (1994). *Similarity for Analogical Software Reuse: A Computational Model*. 11th European Conference on Artificial Intelligence, Amsterdam, The Netherlands, John Wiley & Sons.
- Tessem, B., S. Bjornestad, et al. (1994). ROSA = Reuse of Object-oriented Specifications through Analogy: A Project Framework. Bergen, Department of Information Science, University of Bergen: 23.

THINKING THROUGH DOING

External Representations in Abductive Reasoning

Lorenzo Magnani

Department of Philosophy and Computational Laboratory; University of Pavia; Pavia; Italy
Philosophy, Science, and Technology Program; Georgia Institute of Technology; Atlanta; GA; USA
lmagnani@cc.gatech.edu

Abstract

The concept of *manipulative abduction* is devoted to capture the role of action in many interesting situations: action provides otherwise unavailable information that enables the agent to solve problems by starting and performing a suitable abductive process of generation or selection of hypotheses. Many *external representations*, even if in some cases inert from the epistemological point of view, can be transformed into what is called *epistemic mediators*, active in creative abductive reasoning. I will present some aspects of this kind of reasoning in the case of the discovery of non-Euclidean geometry; moreover, I will illustrate some examples from a computational program that simulates the manipulations of diagrams in geometry.

1 Introduction

What is called *theoretical abduction* (sentential and manipulative) certainly illustrates much of what is important in creative abductive reasoning both in humans and computational programs, especially the objective of selecting and creating a set of hypotheses that are able to dispense good (preferred) explanations of data, but fails to account for many cases of explanations occurring in science or in everyday reasoning when the exploitation of the environment is crucial. The concept of *manipulative abduction* is devoted to capture the role of action in many interesting situations: action provides otherwise unavailable information that enables the agent to solve problems by starting and performing a suitable abductive process of generation or selection of hypotheses.

Many external things, even if usually inert from the epistemological point of view, can be transformed into what is called *epistemic mediators*, which are illustrated in the second part of this paper, together with an analysis of the related notion of “external representation”. I will present some aspects of this kind of reasoning in the case of the discovery of non-Euclidean geometry and I will illustrate some examples from a computational program that simulates the manipulations of diagrams in geometry. The computational embodiment generates in this last case a kind of “squared” epistemic mediator: geometrical construction, as an example of epistemic mediator, is further mediated.

2 Model-based abduction

It is well known that many reasoning conclusions that do not proceed in a deductive manner are *abductive*. What is called *theoretical abduction* (Magnani, 1999a and 2001a) is, from a cognitive point of view, an internal process of reasoning. What about the “external” ways of finding hypotheses?

Many attempts have been made to model abduction by developing some formal tools in order to illustrate its computational properties and the relationships with the different forms of deductive reasoning. Some of the formal models of abductive reasoning are based on the theory of the *epistemic state* of an agent (Boutilier and Becher 1995), where the epistemic state of an individual is modeled as a consistent set of beliefs that can change by expansion and contraction (*belief revision framework*). This kind of *sentential frameworks* exclusively deals with selective abduction¹ (diagnostic reasoning) and relates to the idea of preserving *consistency*. If we want to provide a suitable framework for analyzing the most interesting cases of conceptual changes in science we do not have to limit ourselves to the *sentential* view of theoretical abduction but we have to consider a broader *inferential* one which encompasses both sentential and what is called *model-based* sides of creative abduction.

¹ We have to distinguish between selective and creative abduction. Abduction that merely *selects* from an encyclopedia of pre-stored hypotheses is called selective. Abduction that generates *new* hypotheses (Magnani 1992) is called creative.

Hence, if we want to deal with the nomological and most interesting creative aspects of abduction we are first of all compelled to consider the whole field of the growth of scientific knowledge. Related to the high-level types of scientific conceptual change (Thagard 1992) are different varieties of *model-based abductions* (see, for examples, Magnani 1999b). Following Nersessian (1999), the term “model-based reasoning” is used to indicate the construction and manipulation of various kinds of representations, not necessarily sentential and/or formal. Obvious examples of model-based reasoning are constructing and manipulating visual representations, thought experiment, analogical reasoning, but also the so-called “tunnel effect” (Cornuéjols et al., 2000), occurring when models are built at the intersection of some operational interpretation domain – with its interpretation capabilities – and a new ill-known domain.

What exactly is model-based abduction from a philosophical point of view? Peirce stated that all thinking is in signs, and signs can be icons, indices, or symbols. Moreover, all *inference* is a form of sign activity, where the word sign includes “feeling, image, conception, and other representation” (CP 5.283), and, in Kantian words, all synthetic forms of cognition. That is, a considerable part of the thinking activity is *model-based*. Of course model-based reasoning acquires its peculiar creative relevance when embedded in abductive processes.

For Peirce a Kantian keyword is synthesis, where the intellect constitutes in its forms and in a harmonic way all the material delivered by the senses. Surely Kant did not consider synthesis as a form of *inference* but, notwithstanding the obvious differences, I think synthesis can be related to the Peircian concept of inference, and, consequently, of abduction. After all, when describing the ways the intellect follows to unify and constitute phenomena through imagination Kant itself makes use of the term *rule* (Kant 1929, A140, B179-180, 182), and also of the term *procedure* (Kant 1929, A140-B179-180, 182). We know that rules and procedures represent the central features of the modern concept of inference.

Most of these forms of constitution of phenomena are creative and, moreover, characterized in a model-based way. Let me show some examples of model-based inferences. It is well known the importance Peirce ascribed to diagrammatic thinking, as shown by his discovery of the powerful system of predicate logic based on diagrams or “existential graphs” (Anderson, 1987). As we have already stressed, Peirce considers inferential any cognitive activity whatever, not only conscious abstract thought; he also includes perceptual knowledge and subconscious cognitive activity. For instance in subconscious mental activities visual representations play an immediate role.

Peirce gives an interesting example of model-based abduction (Magnani, 1999a and 2001a) related to sense activity: “A man can distinguish different textures of cloth by feeling: but not immediately, for he requires to move fingers over the cloth, which shows that he is obliged to compare sensations of one instant with those of another” (CP 5.221); this idea surely suggests that abductive movements also have interesting extra-theoretical characteristics and that there is a role in abductive reasoning for various kinds of manipulations of external objects (cf. below, the problem of “action-based, manipulative abduction”). One more example is given by the fact that the perception of tone arises from the activity of the mind only after having noted the rapidity of the vibrations of the sound waves, but the possibility of individuating a tone happens only after having heard several of the sound impulses and after having judged their frequency. Consequently the sensation of pitch is made possible by previous experiences and cognitions stored in memory, so that one oscillation of the air would not produce a tone.

To conclude, all knowing is *inferring* and inferring is not instantaneous, it happens in a process that needs an activity of comparisons involving many kinds of models in a more or less considerable lapse of time. All sensations or perceptions participate in the nature of a unifying hypothesis, that is, in abduction, in the case of emotions too: “Thus the various sounds made by the instruments of the orchestra strike upon the ear, and the result is a peculiar musical emotion, quite distinct from the sounds themselves. This emotion is essentially the same thing as a hypothetical inference, and every hypothetical inference involved the formation of such an emotion” (CP 2.643).

What happens when the abductive reasoning in science is strongly related to extra-theoretical actions and manipulations of “external” objects? When abduction is “action-based” on *external models*? When thinking is “through doing” as illustrated in the simple case above of distinguishing the simple textures of cloth by feeling? To answer these questions I will delineate the first features of what I call *manipulative abduction* by showing how we can find in scientific and everyday reasoning methods of constructivity based on external models and actions.

3 Manipulative abduction

Manipulative abduction happens when we are thinking *through* doing and not only, in a pragmatic sense, about doing. It refers to an extra-theoretical behavior that aims at creating communicable accounts of new experiences to integrate them into previously existing systems of experimental and linguistic (theoretical) practices. Gooding (1990) refers to this kind of concrete manipulative reasoning when he illustrates the role in science of the so-called “construals” that embody tacit inferences in procedures that are often apparatus and machine based. The embodiment is of course an expert

manipulation of objects in a highly constrained experimental environment, and is directed by abductive movements that imply the strategic application of old and new *templates* of behavior mainly connected with extra-theoretical components, for instance emotional, esthetical, ethical, and economic.

The hypothetical character of construals is clear: they can be developed to examine further chances, or discarded, they are provisional creative organization of experience and some of them become in their turn hypothetical *interpretations* of experience, that is more theory-oriented, their reference is gradually stabilized in terms of established observational practices. Step by step the new interpretation - that at the beginning is completely "practice-laden" - relates to more "theoretical" modes of understanding (narrative, visual, diagrammatic, symbolic, conceptual, simulative), closer to the constructive effects of theoretical abduction. When the reference is stabilized the effects of incommensurability with other stabilized observations can become evident. But it is just the construal of certain phenomena that can be shared by the sustainers of rival theories. Gooding (1990) shows how Davy and Faraday could see the same attractive and repulsive actions at work in the phenomena they respectively produced; their discourse and practice as to the role of their construals of phenomena clearly demonstrate they did not inhabit different, incommensurable worlds in some cases. Moreover, the experience is constructed, reconstructed, and distributed across a social network of negotiations among the different scientists by means of construals.

To illustrate this process - from manipulations, to narratives, to possible theoretical models (visual, diagrammatic, symbolic, mathematical) - in a previous work² I have considered some observational techniques and representations made by Faraday, Davy, and Biot concerning Oersted's experiment about electromagnetism. They were able to create consensus because of their conjectural representations that enabled them to resolve phenomena into stable perceptual experiences. Some of these narratives are very interesting.

3.1 Epistemic mediators

Recent research, taking an ecological approach to the analysis and design of human-machine systems, has shown how expert performers use action in everyday life to create an *external* model of task dynamics that can be used in lieu of an internal model (Kirlik, 1998). Not only a way for moving the world to desirable states, action performs an *epistemic* and not merely performatory role that is very relevant to abductive reasoning.

² Cf. Magnani, 2001 (also studied in Gooding, 1990).

The whole activity of manipulation is devoted to build various external *epistemic mediators* that function as an enormous new source of information and knowledge. I derive this expression from the cognitive anthropologist Hutchins (1995), that coins the expression "mediating structure" to refer to various external tools that can be built to cognitively help the activity of navigating in modern but also in "primitive" settings. Any written procedure is a simple example of a cognitive "mediating structure" with possible cognitive aims: "Language, cultural knowledge, mental models, arithmetic procedures, and rules of logic are all mediating structures too. So are traffic lights, supermarkets layouts, and the contexts we arrange for one another's behavior. Mediating structures can be embodied in artifacts, in ideas, in systems of social interactions [...]" (290-291).

In this light manipulative abduction in science represents a kind of redistribution of the epistemic and cognitive effort to manage objects and information that cannot be immediately represented or found internally (for example exploiting the resources of visual imagery).³

3.2 Experiments and the "World of Paper"

Already in the *Dialogues Concerning the Two Chief World Systems* (1632), accentuating the role of observational manipulations Galileo presents an anatomist that, manipulating a cadaver, is able to get new, not speculative, information that goes beyond the "world of paper" of the Peripatetic philosophy. It is well known that recent philosophy of science has paid a great attention to the so-called theory-ladenness of scientific facts (Hanson, Popper, Kuhn) (Chalmers, 1999). Nevertheless a lot of new information in science is reached by observations and experiments, and experiments are the fruit of various kinds of artifactual manipulations: the different strategies correspond to the expert manipulations of objects in a highly constrained experimental environment, directed by *abductive* movements that imply the application of old and new extra-theoretical *templates* (cf. the following section) of behavior.

With Galileo's achievements, we observe that human "scientific" thinking is related to the manipulation of a material and experimental environment that is no longer natural. Knowledge is finally seen as something cognitively distributed across scientists, their internal "minds", and external artifacts and instruments. Experiments and instruments embody in their turn external crystallization of knowledge and practice. Modern science is made by this interplay of internal and exter-

³ It is difficult to preserve precise spatial relationships using mental imagery, especially when one set of them has to be moved relative to another.

nal. An immediate consequence of Galileo's ideas is the critique of the authority, that advocated the knowledge relevance of a "world of paper", mainly internal from the cognitive point of view. Gooding observes: "It is ironical that while many philosophers admire science because it is empirical as well as rational, philosophical practice confines it to the literary view that Galileo rejected" (1990, xii). Galileo's "book of nature" and his systematic use of the telescope are the revolutionary *epistemic mediators* that characterize the cognitive power of the new way of producing intelligibility.

3.3 Manipulative templates

We still know very little about what governs the action-based abduction. I plan to better delineate some of the manipulative *templates*⁴ that are active in creative abduction comparing scientific and everyday reasoning: 1. action elaborates a *simplification* of the reasoning task and a redistribution of effort across time when we "need to manipulate concrete things in order to understand structures which are otherwise too abstract" (Piaget 1974), or when we are in presence of *redundant* and unmanageable information; 2. action can be useful in presence of *incomplete* or *inconsistent* information - not only from the "perceptual" point of view - or of a diminished capacity to act upon the world: it is used to get more data to restore coherence and to improve deficient knowledge; 3. action as a *control of sense data* illustrates how we can change the position of our body (and/or of the external objects) and how to exploit various kinds of prostheses (Galileo's telescope, technological instruments and interfaces) to get various new kinds of stimulation: action provides some tactile and visual information (e. g., in surgery), otherwise unavailable; 4. action enables us to build *external artificial models* of task mechanisms instead of the corresponding internal ones, that are adequate to adapt the environment to agent's needs: experimental manipulations exploit *artificial apparatus* to free new possible stable and repeatable sources of information about hidden knowledge and constraints; 5. in science experimental action shows a sensibility to the aspects of the phenomenon which can be regarded as *curious* or *anomalous*; manipulations have to be able to introduce potential inconsistencies in the received knowledge (Oersted's report of his well-known experiment about electromagnetism is devoted to describe some anomalous aspects that did not depend on any particular theory of the nature of electricity and magnetism; 6. action exhibits a preliminary sensibility to the *dynamical* character of the phenomenon, and not to entities and their properties, common aim of manipulations is to practically reorder the dynamic sequence of events in a static spatial one that should promote a subsequent bird's-eye view (narrative or visual-diagrammatic).

⁴ As a kind of schematic and general habit typical of many epistemological and/or cognitive behaviors.

4 Geometrical construction is a kind of manipulative abduction

It is well-known that in the history of geometry many researchers used internal mental imagery and mental representations of diagrams, but also self-generated diagrams (external) to help their thinking (Otte and Panza, 1997). For example, it is clear that in geometrical construction many of the requirements indicated by the manipulative templates (cf. above) are fulfilled. Indeed geometrical constructions present situations that are curious and "at the limit". They are constitutively dynamic, artificial, and offer various contingent ways of epistemic acting, like looking from different perspectives, comparing subsequent appearances, discarding, choosing, re-ordering, and evaluating. Moreover, they present the features typical of manipulative reasoning illustrated above, such as the simplification of the task and the capacity to get visual information otherwise unavailable.

Let's quote an interesting Peirce's passage about constructions. Peirce says that mathematical and geometrical reasoning "consists in constructing a diagram according to a general precept⁵, in observing certain relations between parts of that diagram not explicitly required by the precept, showing that these relations will hold for all such diagrams, and in formulating this conclusion in general terms. All valid necessary reasoning is in fact thus diagrammatic" (CP, 1.54). Not dissimilarly Kant says that in geometrical construction "[...] I must not restrict my attention to what I am actually thinking in my concept of a triangle (this is nothing more than the mere definition); I must pass beyond it to properties which are not contained in this concept, but yet belong to it" (Kant, 1929, A718-B746, p. 580).

We have seen that manipulative abduction is a kind of abduction, usually model-based, that exploits external models endowed with delegated (and often implicit) cognitive roles and attributes.

- The model (diagram) is *external* and the strategy that organizes the manipulations is unknown *a priori*.
- The result achieved is *new* (if we, for instance, refer to the constructions of the first creators of geometry), and adds properties not contained before in the concept (the Kantian to "pass beyond" or "advance beyond" the given concept, Kant, 1929, A154-B193/194, p. 192).⁶

⁵ That is a kind of definition that prescribes "what you are to *do* in order to gain perceptual acquaintance with the object of the world" (CP, 2.330).

⁶ Of course in the case we are using diagrams to demonstrate already known theorems (for instance in didactic settings), the strategy of manipulations is not necessary unknown and the result is not new.

Hence, in the construction of mathematical concepts many external representations are exploited, both in terms of diagrams and of symbols. I am interested in my research in the diagrams which play an *optical* role - microscopes (that look at the infinitesimally small details), telescopes (that look at infinity), windows (that look at particularly situation), a *mirror* role (to externalize rough mental models), and an *unveiling* role (to help to create new and interesting mathematical concepts, theories, and structures). I describe them as the *epistemic mediators* (cf. above) able to perform various abductive tasks (discovery of new properties or new propositions/hypotheses, provision of suitable sequences of models able to convincingly verifying theorems, etc.). Elsewhere I have presented some details concerning the role of optical diagrams in the calculus (Magnani and Dossena, 2002).

5 Discovering new concepts with diagrams and mechanizing manipulative abduction

It is clear that humans and other animals make a great use of perceptual reasoning and kinesthetic abilities. We can catch a thrown ball, cross a busy street, read a musical score, go through a passage by imaging if we can contort out bodies to the way required, evaluate shape by touch, recognize that an obscurely seen face belongs to a friend of ours, etc. Usually the “computations” required to achieve these tasks are not accessible to a conscious description. Mathematical reasoning uses language explanations, but also non-linguistic notational devices and models. Geometrical constructions represent a relatively simple example of this kind of extra-linguistic machinery we know as characterized in a model-based and manipulative - abductive - way.

5.1 Unveiling and mirror diagrams in the discovery of geometrical concepts

Diagrams serve an important role in abduction because they can be manipulated. In mathematics diagrams play various roles in a typical abductive way. Two of them are central:

- they provide an intuitive *explanation* able to help the understanding of concepts difficult to grasp or that appear obscure and/or epistemologically unjustified,⁷
- they help to *create* new previous unknown concepts, as illustrated in the case of the non-Euclidean geometry.

⁷ Some new optical diagrams (microscopes within microscopes), which provide new mental representations of the concept of tangent line at the infinitesimally small regions, are introduced in the already cited Magnani and Dossena (2002).

In the case of the construction and examination of diagrams in geometrical reasoning, specific experiments serve as states and the implied operators are the manipulations and observations that transform one state into another. The geometrical outcome is dependent upon practices and specific sensory-motor activities performed on a non symbolic object, which acts as a dedicated external representational medium supporting the various operators at work. There is a kind of an epistemic negotiation between the sensory framework of the geometer and the external reality of the diagram. This process involves an *external representation* consisting of written symbols and figures that are manipulated “by hand”. The cognitive system is not merely the mind-brain of the person performing the geometrical task, but the system consisting of the whole body (cognition is *embodied*) of the person plus the external physical representation. In geometrical discovery the whole activity of cognition is located in the system consisting of a human together with diagrams.

An external representation can modify the kind of computation that a human agent uses to reason about a problem: the Roman numeration system eliminates, by means of the external signs, some of the hardest parts of the addition, whereas the Arabic system does the same in the case of the difficult computations in multiplication (Zhang, 1997).

The external representations are not merely memory aids: they can give people access to knowledge and skills that are unavailable to internal representations, help researchers to easily identify aspects and to make further inferences, they constrain the range of possible cognitive outcomes in a way that some actions are allowed and other forbidden. The mind is limited because of the restricted range of information processing, the limited power of working memory and attention, the limited speed of some learning and reasoning operations; on the other hand the environment is intricate, because of the huge amount of data, real time requirement, uncertainty factors. Consequently, we have to consider the whole system, consisting of both internal and external representations, and their role in optimizing the whole cognitive performance of the distribution of the various subtasks (Trafton et. al., 2002). We already stated that in the history of geometry many researchers used internal mental imagery and mental representations of diagrams, but also self-generated diagrams (external) to help their thinking.

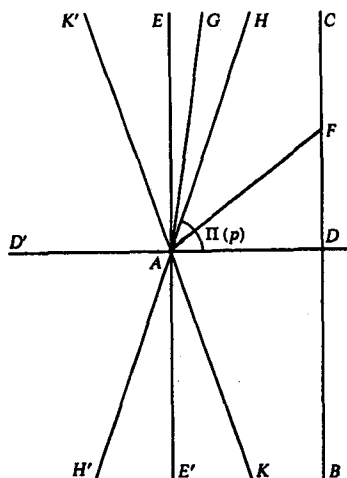
Mirror and unveiling diagrams play a fundamental creative and explanatory (sometimes also didactic) role to remove obstacles and obscurities and to enhance mathematical knowledge of critical situations. They facilitate new internal representations and new symbolic-propositional achievements. The unveiling diagrams provide new light on mathematical structures: these diagrams can lead to interesting creative results. Finally, the mirror and unveiling diagrammatic representations of mathematical structures activate *direct*

perceptual operations (for example how to identify where we can find an orisphere in an Euclidean structure; how to represent a stereometric non-Euclidean form, cf. below).

We stated above that in mathematics mirror and unveiling diagrams play various roles in a typical abductive way. Now we can add that:

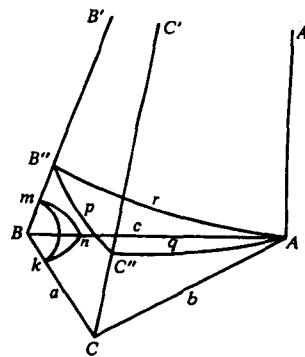
- they are *epistemic mediators* able to perform various abductive tasks in so far as
- they are *external representations* which, in the cases we will present in the following sections, are devoted to provide explanatory abductive results.

Figure 1.



Let us consider some aspects of the role of mirror and unveiling diagrams in the Lobachevskyan discovery of the elementary non-Euclidean geometry. A *mirror diagram* (for example the diagram of the drawn parallel lines - cf. Figure 1 - Lobachevsky, 1891) is a kind of external *analogous* both of the mental image we depict in the mental visual buffer and of the symbolic-propositional level of the postulate definition (the fifth postulate). In general this diagram mirrors the internal imagery and provide the possibility of detecting anomalies. The external representation of geometrical structures often activates direct perceptual operations (for example, identify the parallels and search for the limits) to elicit consistency or inconsistency routines. Sometimes the mirror diagram biases are inconsistent with the task and so they can make the task more difficult by misguiding actions away from the goal. If consistent, they can make the task easier by guiding actions toward the goal. In certain cases the mirror diagrams biases are irrelevant, they should have no effects on the decision of abductive actions, and play lower cognitive roles. The in the case of Figure 1 the diagram of the parallel lines was used in the history of geometry to make both consistent and inconsistent the fifth Euclidean postulate and the new non-Euclidean perspective (more details on this epistemological situation are given in Magnani, 2001b).

Figure 2.



An example of *unveiling diagram* is the one illustrated by the Figure 2 (Lobachevsky, 1891). It is more abstract than the previous one and exploits “audacious” representations in the perspective of three dimensional geometrical shapes. The construction given in the figure aims at diagrammatically “representing” a stereometric non-Euclidean form built on a rectilinear right angled triangle ABC to which theorems previously proved (for example, the one stating that the parallels AA' , BB' , CC' , which lie on the three planes are parallels in non-Euclidean sense) can be applied. In this way Lobachevsky is able to further apply symbolic identifications and to arrive to new equations which consistently (and in the same time) connect Euclidean and non-Euclidean perspectives. This kind of diagram strongly guides the geometer’s selections of moves by eliciting what I call the *Euclidean-inside non-Euclidean* “model matching strategy”. This maneuver also constitutes an important step in the affirmation of the modern “scientific” concept of model. This unveiling diagram constitutes a kind of gateway to imaginary entities.

In general we have to note that some perceptions activated by the diagram are of course disregarded as irrelevant to the task, as it usually happens when exploiting external diagrammatic representations in reasoning processes. Because not everything in external representations is always relevant to a task, high level cognitive mechanisms need to use task knowledge (usually supplied by task instructions, geometrical, in our case) to direct attention and perceptual processes to the relevant features of external representations. This external representation in terms of an unveiling diagram activates a perceptual reorientation in the construction (that is identifies possible further constructions and enhancements); in the meantime the generated internal representation of the external elements activates directly retrievable information (numerical values) that elicits the strategy of building further non-Euclidean structures together with their *analytic* counterpart (the non-Euclidean trigonometry equations)⁸.

⁸ More details concerning the role of mirror and unveiling diagrams in constructing different ideas of infinity in Lobachevsky’s thought are given in Magnani, 2002).

5.1 Automatic geometrical constructions as epistemic mediators

A very interesting artificial intelligence computer program has been built, ARCHIMEDES (Lindsay, 1994, 1998), that represents geometrical diagrams (points, line segments, polygons, and circles) both as pixels arrays and as propositional statements. For example a triangle will be represented from a propositional description as a set of marked pixels in an array, together with a set of data naming the given triangle and storing facts about it (for instance that it is right) and constraints upon it (perhaps that it remains right throughout this use of the diagram). Hence, a computational equivalent of a physical diagram is represented, plus some human propositional knowledge about it.

The program is able to “manipulate” and modify its own representations of diagrams, that is it is able to make *geometrical constructions* (called “simulation constructions”): adding parts or elements, moving components about, translating and rotating by preserving metric properties, of course subordinated to the given specific constraints and to the whole structure of the two-dimensional space. Some knowledge of algebra is added, and of the taxonomic hierarchy of geometric figures (all square are rectangles, etc.); moreover, it is also added additional knowledge like side-angle-side congruency theorem and the sum of the interior angles of a triangle, knowledge of problem solving strategies and heuristics, knowledge of logic (for example: a universal statement can be disproved by a single counterexample) (Lindsay, 2000b).

When the program manipulates the specific diagram, it records the new information that comes out, then it can for example detect sets of area equivalences, and so on: for example, it is able to verify that a demonstration of the Pythagorean Theorem is correct, mirroring its truth in terms of constructions and manipulations. To account for the universality of geometrical theorems and propositions many different methods for learning and “generalizing” the specific instance of the constructed diagram are exploited (Lindsay, 1988, pp. 260-264).

These methods come from a kind of predicative knowledge of course “exogenous” to the mere diagrammatic representation: generalization is not a possible product of the pure diagrammatic understanding. For instance, one suggestion is to break the problem into cases, to individuate a “representative” instance for each case, and to demonstrate that this conclusion holds for each of these instances. Another one is to exploit the simulative aspects of constructions by “running experiments” that show how some parts of a diagram co-vary with changes in others: the observation of the interaction of the diagrams parts as one property is varied allows us to grasp and understand the “universal” value of some geometric relations and results (for example congruency theorem, asymptotic behaviors,

periodic relations, and some symmetric relations). It is interesting to note that one of the construction manipulations proposed by the program to verify the Pythagorean Theorem intends to show that it is not true of (some examples of) non-right triangles.

An extension of the program (described in Lindsay, 2000a), aims to autonomously build constructions that can demonstrate a given proposition,⁹ rather than simply verify them. It accomplishes the further complex task of discovering conjectures that can lead to the constructions of demonstrations, illustrating the possible role of diagrammatic reasoning in creativity.

6 Conclusion

The concepts of model-based and manipulative abduction we have described provide a better understanding of many kinds of reasoning based on external representations and their treatment.

There are many creative and didactic settings where some mechanisms underlying these epistemic mediators can be studied and elicited. For example: i) the role of *optical* diagrams both in the calculus¹⁰ seems relevant.

I am preparing a research devoted to detect the details of their didactic effects on the calculus students at the University of Pavia (mathematics and engineering curricula and teaching environment); I am also convinced these kinds of diagrams can be exploited and studied as epistemic mediators in everyday non-mathematical applications (finding routes, road signs, buildings maps, for example), in connection with various zooming effects of spatial reasoning; ii) I think the cognitive activities of optical, mirror, and unveiling diagrams can be studied in other areas of model-based reasoning (Magnani and Nersessian, 2002), such as the ones involving creative, analogical, and spatial inferences, both in science and everyday situations so that this can extend the epistemological and the psychological theory.

References

- D.R., Anderson. *Creativity and the Philosophy of Charles Sanders Peirce*. Clarendon Press, Oxford, 1987.
- C. Boutilier and V. Becher. Abduction as belief revision. *Artificial intelligence*, 77:43-94, 1995.

⁹ The program is able to “discover demonstrations”, that is to find sequences of manipulations that achieve a particular end.

¹⁰ Cf. footnote 7.

- A.F. Chalmers. *What is this Thing Called Science* (1976). Hackett, Indianapolis/Cambridge, 1999.
- A. Cornuéjols, A. Tiberghien and G. Collet. A new mechanism for transfer between conceptual domains in scientific discovery and education. *Foundations of Science*, 5(2):129-155, 2000. Special Issue on "Model-based Reasoning in Science: Learning and Discovery", ed. by L. Magnani, N.J. Nersessian, and P. Thagard.
- G. Galilei. *Dialogues Concerning the Two Chief World Systems* (1632), translated by S. Drake. In: M.R. Matthews, ed., 1989, pp. 61-81.
- D. Gooding. *Experiment and the Making of Meaning*. Kluwer, Dordrecht, 1990.
- E. Hutchins. *Cognition in the Wild*, MIT Press, Cambridge, MA, 1995.
- I. Kant. *Critique of Pure Reason* (1787), translated by N. Kemp Smith. MacMillan, London, 1969, reprint 1998.
- A. Kirlik. The ecological expert: acting to create information to guide action. In: *Proceedings of the 1998 Conference on Human Interaction with Complex Systems* (HICS'98), IEEE Press, Piscataway, NJ, 1998.
- R.K. Lindsay. Understanding diagrammatic demonstrations. In: *Proceedings of the 16th Annual Conference of the Cognitive Science Society*, A. Ram and K. Eiselt, eds., Erlbaum, Hillsdale, NJ, 1994, pp. 572-576.
- R.K. Lindsay. Using diagrams to understand geometry, *Computational Intelligence*, 9(4):343-345, 1998.
- R.K. Lindsay. Using spatial semantics to discover and verify diagrammatic demonstrations of geometric propositions. In: *Spatial Cognition*, S. O'Niallan, ed., John Benjamins, Amsterdam, 2000a, pp. 199-212.
- R.K. Lindsay. Playing with diagrams. In: *Diagrams 2000*, M. Anderson, P. Cheng, and V. Haarslev, eds., Springer, Berlin, 2000b.
- R.K. Lindsay. Knowing about diagrams. In: *Reasoning with Diagrams*, M. Anderson and P. Olivier, eds., Springer, Berlin, 2000c.
- N.J. Lobachevsky. *Geometrical Researches on the Theory of Paralleles* [1840], translated by G.B. Halsted. University of Texas, Austin, 1891.
- L. Magnani. Abductive reasoning: philosophical and educational perspectives in medicine. In D. A. Evans and V. L. Patel (eds.). *Advanced Models of Cognition for Medical Training and Practice*. Springer, Berlin, 1992, pp. 21-41.
- L. Magnani. Model-based creative abduction. In: L. Magnani, N. J. Nersessian, and P. Thagard, eds., *Model-Based Reasoning in Scientific Discovery*, Kluwer Academic/Plenum Publishers, New York, 1999a, pp. 219-238.
- L. Magnani, Inconsistencies and Creative Abduction in Science. In: *AI and Scientific Creativity. Proceedings of the AISB99 Symposium on Scientific Creativity*, Society for the Study of Artificial Intelligence and Simulation of Behaviour, University of Edinburgh, Edinburgh, 1999b, pp. 1-8.
- L. Magnani. *Abduction, Reason, and Science. Processes of Discovery and Explanation*. Kluwer Academic/Plenum Publishers, New York, 2001a.
- L. Magnani. *Philosophy and Geometry. Theoretical and Historical Issues*. Kluwer Academic, Dordrecht, 2001b.
- L. Magnani. Epistemic mediators and model-based discovery in science, in: L. Magnani and N.J. Nersessian, eds, 2002, pp. 305-329.
- L. Magnani and R. Dossena. Perceiving the infinite and the infinitesimal world: unveiling and optical diagrams and the construction of mathematical concepts, submitted to CogSci2002.
- L. Magnani and N.J. Nersessian, eds. *Model-Based Reasoning: Science, Technology, Values*. Kluwer Academic/Plenum Publishers, New York, 2002.
- M.R. Matthews. *The Scientific Background to Modern Philosophy*. Hackett, Indianapolis/Cambridge, 1989.
- N.J. Nersessian. Model-based reasoning in conceptual change. In: L. Magnani, N. J. Nersessian, and P. Thagard eds., *Model-Based Reasoning in Scientific Discovery*, Kluwer Academic/Plenum Publishers, New York, 1999, pp. 5-22.
- M. Otte and M. Panza, eds. *Analysis and Synthesis in Mathematics*. Kluwer Academic, Dordrecht, 1997.
- C.S. Peirce. *Collected Papers* 1-6 (CP), edited by C. Hartshorne and P. Weiss. *Collected Papers* 7-8, edited by A. Burks, Harvard University Press, Cambridge, 1931-35, 1958.
- J. Piaget, *Adaptation and Intelligence*, University of Chicago Press, Chicago, IL, 1974.
- P. Thagard. *Conceptual Revolutions*. Princeton University Press, Princeton, 1992.

Intelligent Dynamic Collaboration

Kirsty A. Beilharz

Key Centre of Design Computing and Cognition

University of Sydney, Australia

kirsty@arch.usyd.edu.au

Abstract

This paper investigates criteria required for developing a robotic or computational collaborative tool for designers / creators. The intention of this tool is a real time collaborative assistant or collaborator that provides innovative designs in reaction to, and adaptation of, material in a situated design context. The human designer provides the cathartic stimuli to which the collaborative tool responds and using adaptive, reactive and learned knowledge the collaborator responds with material in response and adaptation to the constantly evolving, hence dynamic, situated design context. As the human designers ideas and formulations evolve and morph, the computational or robotic collaborator model adapts to the transforming dominant trends in creativity to shape the material it produces in response. This model is not designed to replace or simulate human creativity; rather it is a productive, responsive complementary tool, complicit to the creative tendencies of the human design developer. This is a concept prototype for which an hypothetical design scenario is considered in order to address formation of hierarchy and categorisation of design parameters that are situated and specific to the design context. Essential to a real time reactive agent or computational collaborative model, is its ability to learn and to discard knowledge from past, learned, situated experiences as the situation changes with the dynamic of the human designer. It is envisaged that this tool is useful at those stages of the designing process in which the productivity of the principal (human) designer is relatively rapid, intuitive and spontaneous. Its potential benefit, therefore, is at the conceptual stage of design or in a situation in which output is delivered immediately, e.g. an improvisatory or design-screening stage in the design process.

1 Introduction

1.1 Terminology

In this paper, “designer” is used to refer to the human designer or creator who provides the initial and transforming information and material to which the collaborative tool responds and reacts. Although both designer and collaborative assistant are creative in generating new design material, the collaborator continuously produces its material in reaction to the human designer.

In this work “design” is viewed as

“a situated and dynamic activity”. (Reffat & Gero, 2000)

The term “collaborator” is used to refer to the collaborative, also generative or creative, computational design assistant but the innovation and adaptation of material produced by the collaborator is in response to the dynamic situation provided by the designer.

Throughout this paper, it is assumed that the design process and the creative material it generates are inseparable from its context. The process of designing is conditioned by what has preceded it, both in the case of the human designer and in the case of the computational collaborator. On the part of the collaborative tool, the material it generates is further conditioned by reaction to the designers trends and production. Due to the reactive and

adaptive nature of the material produced by the collaborator, the model is entirely dependent on the output of the designer. As the designers concepts evolve and transform, so too the collaborators reactions evolve and learn from situated experiences. The process of evolving and discerning principal tendencies and pursuing their change by the collaborator renders the model irremovable from its context.

Designers actions are situation dependent, i.e. situated, such that designers work interactively with the design environment within the specific conditions of the situation. (Reffat & Gero, 2000; Gedenryd, 1998)

Compliant with this observation of the design process, this model is situation dependent and, especially the role of the collaborator is wholly situated according to the actions of the designer with whom the collaborative tool is interacting. This notion of the collaborators response is stretched to include not only categories of “reactive” and “adaptive” response, but also to term it “interactive” because the material generated by the collaborator is immediately returned to the designer who may react to it and thence the material is further reinterpolated by both designer and collaborator as the design continues to evolve.

A “reactive” response is one in which the collaborator provides a simple response in answer, reciprocation or response to the material it receives. No computational transformation of context or material is required in this

reaction. An example of this response is if the collaborator establishes that no change in dynamic or deviation from its learned knowledge about the material is required before it provides a contextual response.

By comparison, an “adaptive” response is one in which the collaborator senses a change in the dynamic conditions of the design or to which the collaborator finds no appropriate response in its learned knowledge for response.

In this instance, the collaborator must transform material to provide a response derived from the knowledge it has learned not of the design material but about the design process, about the way in which the designer thinks or operates. The response it provides is interpolated according to rules of transformation observed from the designer then applied to the material it has received while comparing it against other conditions in the current design context to ensure it is compatible at that moment in time, given that some parameters of design may change at different rates to others. This complex procedure requires computational generation concurrently with observation.

It assumes that the parameters of observation have been established to accommodate the most important criteria for constancy and change.

The collaborator also needs to be able to adapt to the changing hierarchy of determining parameters, as the dominating combination of important criteria is certain to be dynamic in its combination and parts.

N.B. Since the expression “dynamic” here is used in reference to the changing, mobile nature of design environments and the collaborative agents ability to adapt to a dynamic environment, when subsequent discussion refers to the dynamic level of sound or music it is always referred to as “dynamic (loudness)” for differentiation.

1.2 Significance

Importantly, the design situation of a human and collaborative reaction must occur in real time and immediately because the material generated by the designer is continually transforming in the order, hierarchy and selectivity of its components but the composite result of designer and collaborator is perceived or delivered concurrently, simultaneously.

The significance of a real time collaborator is threefold. Firstly, the collaborator does not replace, replicate or mimic the designer. It has creative or generative capabilities of its own which are interdependent on the designer. It maintains an intelligent but subservient role in the design process while designing new material in a situated environment. Secondly, by providing rapid response, the collaborator has the potential to complement, accompany or augment and increase the rapidity of the design process while retaining the design values of the principal decision-maker. Thirdly, it has the ability to change with the dynamic of the design in a manner comparable to human designing thought by constructing situated knowledge, not only about the designed material but also about

the designers methodology.

It is significant that a considerable amount of research has been conducted into computer generative creativity, i.e. in which the autonomous computer or agent innovates in place of the designer or for the designer, both in architectural designing and in musical composition. Tools also exist for pre-programmed accompaniment but this concept is novel in its ambition to both innovate and observe simultaneously while learning from a dynamic source outside its initial programmed knowledge.

1.3 Implementation Scenarios

The hypothetical scenarios in which this collaborative tool could be implemented are insignificant or unspecific in that the concept for this model could be applied to various dynamic real time creative situations. The scenario are considered here, for developing a methodology for addressing the dynamic nature of the source material produced by the designer and to formulate a methodology for prioritising, categorising and reacting to this dynamic, situated material on the part of the collaborator.

Whether the implementation of the collaborative tool is computational, mechanical or robotic, is purely speculative because the nature of the output by the collaborator is completely dependent on the design situation. To this end, the two scenarios to be considered, posed for the purpose of formalising the model, explained subsequently, is a live performance musical improvisatory collaboration.

Hypothetically, there are other design instances in which the collaborative tool could prove its usefulness by increasing productivity and rapidity of producing design alternatives for consideration. Most design tools require the user to make selections from machine generated designs, whereas this model applies the intelligence of designer-determined criteria, selectivity and preferences to generate material that is highly likely to conform to the sensibilities of the designer because his/her material and methods of creativity are the source for the collaborators innovation. An example of an alternative scenario is utilising the collaborator to generate a series of rendering alternatives for an architectural model design in order to canvas opinion from the prospective client. The alternatives provided may be “guided” or tempered according to guidelines followed by the designer in response to the client.

In the situation proposed in this paper, the medium of sound and live performance tests the responsiveness and adaptive capabilities of the collaborator tool because the outcome is delivered in real time, almost simultaneously, and the sound produced, i.e. the creative outcome, is perceived immediately and concurrently by the (third-party) audience. Typically, in most existing collaborative musical tools, the human must be complicit to the accompanying material because the assistant has no adaptive or reactive intelligence with which to formulate new material in a dynamic situation.

An example of this scenario is a Jazz collaborator that must adapt to unknown factors of an already improvising musician / designer. Musically, the designer will follow certain rules and establish certain patterns of behaviour (e.g. reiterate rhythm cells or frequently utilised pitch patterns) but criteria such as transposition and key, or rhythmic pulse and tempo, may change and the collaborator must learn the designers / improvisers methods for innovation by observing how material is transformed. The material it produces in response occurs in real time changing according to the designers context. Wiggins (Curry & Wiggins, 1999; Papadopoulos & Wiggins, 1998) and Johnson-Laird (HREF1; HREF3) both refer to, or have modelled, Jazz improvising or composing computers in which the program creates new interpolations of melody and other parameters according to a grammatical construction based on Jazz music practice. This is quite different from the function of the collaborator in this paper, which must learn its creative procedures and determine relevant parameters in which to operate from a dynamic design environment determined transiently outside the computer program.

This collaborative design scenario is explored because the author is familiar with this scenario and has conducted considerable analysis into the process of musical "live" or real time response. It is important to remember, however, that this is purely an example for illustration purposes and that the concept for the collaborator tool can be applied to many situations, providing that it is appropriately programmed. Its significance as a concept for intelligent collaboration is that the model is adaptable and configurable.

2 Collaborative Intelligence

2.1 Intelligent Reaction in a Dynamic Environment

Designing, and knowledge required for designing, depend on situation. In order for the collaborator to be able to react in the ways described following, it must first be able to learn from, and "understand", a dynamic environment. This means that the designer who is responsible for sending the data that triggers the reactions of the collaborator is sending information the nature of which, and for which the context, constantly changes.

The first part of the collaborators intelligence is concerned with assessing and responding to the changing importance of criteria or parameters of design. In order to do this, the collaborator must analyse and impose hierarchy on the observed material sourced from the principal designer. The analysis process is considered in 3.1. The aspects of design that change in a dynamic creative environment are the composition of elements, the constituent parameters, and the importance of those parameters relative to one another. The collaborator is learning two main strands of knowledge: knowledge of designed material and knowledge of the design process observed in the hu-

man designer. Both strands change throughout the creative process.

If the collaborator were simply to accrue all the knowledge directed towards it, it would eventually overload with data, much of it superfluous and no longer relevant. The dynamic collaborative agent therefore must discern the principals of design creativity and material that are temporally dominant or contextual according to its knowledge. This constitutes situated knowledge and must be determined by a series of filters.

The filters determining relevance can be based on currency and proliferation, i.e. those design practices utilised constantly and currently must be retained and operated with. Those that have not been utilised recently (the constraints of which are determined by storage capacity and computational power) can be deemed obsolete and may be discarded. This is computationally derived from incidence frequency and locality. To use the example of the musical improviser scenario, those pitch sequences that are reiterated frequently and remain connected to the currently relevant key centre are retained while those used extremely infrequently and not recently, may be discarded and may be relearned if the dynamic situation subsequently requires it.

The learned knowledge and behaviours useful if retained are likely to be data pertaining to the designers actual practice of creativity and innovation, i.e. if the collaborator retains and develops knowledge about how the designer thinks and creates, human thought patterns are less likely to change at the same rate as the material on which the permutations themselves are performed. This is based on the premise that what we call "style" is a form of design grammar. Grammar characterises individuals while the material with which designers work changes rapidly. Meanwhile, the collaborator must remain dynamic and sensitive to modifications in grammar. If the computational collaborator operated only with initially learned grammars, its habits would evolve independently and in obsolescence from the designer who, as the creative human in this equation, holds the "right" or "authority" to change. For this knowledge analysis to be effective, its accuracy is distinguished by the parameters and criteria it is programmed to process. Hence, the decisions about the relevance and methods of analysis are critical to the success of the collaborator. This is discussed in 3.1.

The process of intelligent reaction in a dynamic environment is therefore concerned with choice of relevant response criteria and elimination of redundancy. This action of choice needs to be adaptable and changing. This means that the enclave of knowledge utilised in reaction and adaptation is a module of dynamic composition. This is the key to a collaborator able to evolve and change in real time alongside a human designer.

2.2 Intelligent Learning from the Designers Creative Practice

Intelligent learning from the designers creative practice is essential to provide the collaborator with methodologies for tackling unfamiliar or novel situations in which an adaptive response is required.

In the case that a simple reactive response is inadequate, unsatisfactory or unavailable, the computational collaborator must be able to innovate a response “in character” or “appropriate” to the dynamic situation. This involves a combination of knowledge of grammar and knowledge learned through the situation characterising the methods the designer uses to invent. The collaborator is called upon to invent or design using these methodologies learned from the designer to produce a coherent and situated design response of its own. Because the collaborator is implementing design grammar learned from the designer, aesthetically its outcome is likely to be compatible with the concurrent generation by the designer himself/herself. This understanding of learned grammar is compared against currently operative conditions in the design environment and any incompatibility is eliminated.

For example, if the collaborator observes that chromaticism has been completely and methodically obliterated by the designer in certain keys, it may learn by comparison that, although this may be a grammatical product variable, it is, according to probability, inappropriate to the design perception of the human controller.

If, for instance, the musical improvising collaborator learns that whole-tone adjacent pitch steps are characteristic grammar of the designer/improvising performer but compares this information against the currently relevant key and finds the two are incompatible, the whole-tone step pattern can not be used and a substitute must be selected or invented according to different relevant grammatical tendencies.

3 Response Criteria, Reaction and Adaptation

3.1 Selecting and Prioritising Response Criteria for the Collaborator

In this model, the quality of decisions relating to situated knowledge is dependent on the criteria set for intelligent response in the collaborator and its computational ability to adapt to change. The success of the entire process therefore hinges on dynamic prioritisation of criteria and the programmed and learned criteria used by the collaborator. The collaborator's knowledge about design creativity, relevance and incidence of criteria, and parameters is dynamic but the analysis tools of the collaborator must initially be determined. The initial human programming of computational tool determines the selection of active knowledge filters.

In the same way that a German history book about World War II reads differently to an English one, or that a lions view of a hare differs to the hares view of the lion, the “intelligence” and “interpretative” abilities of the collaborative agent depend on the tools for knowledge accruelement and analysis at its disposal. It is critical which parameters should be measured and any known inherent hierarchy that is unchanging, e.g. in Western music, for an improvising collaborator, the parameters of pitch and rhythm would likely be given precedence over timbre and register. The scope and constraint of the filters also needs to be determined, e.g. the degree of inflection or flexibility of pitch permitted before a pitch is deemed to be another, i.e. the scope of definition and constraints in harmonisation might include the degree of permissible dissonance considered acceptable to the designer.

Given the aesthetic and subjective nature of defining and constraining these parameters for perception by the collaborator, it is suggested that the principal designer is involved in the programming process shaping the outcome of his/her collaborative tool. In this way, the collaborator may “learn” individual sensitivities from the designer or the designers individual views of perception may be programmed according to preferences governing the behaviour of the collaborator. This suggests that the collaborator needs to be customisable by the designer. This customisation both optimises performance and is crucial to the flexibility of the model in its application, as per 1.3 in the Introduction to this paper.

The number, type, specification, scope, restriction of parameters or categories of classification, as well as sensitivity, adaptability, expansiveness, distinguish the intelligent collaborator from existing collaborative devices, e.g. in the field of musical accompaniment, a simple (pitch) harmoniser equivalent or (rhythmic) drum machine. The comparative simplicity of these existing tools means that the harmoniser is an unintelligent, signal-processing device that merely produces a filtered version of the performers/designers output (i.e. requiring no creativity or invention on the part of the machine). The drum machine plays a pre-programmed sequence of patterns to which the performer /designer must then conform without scope for deviation or innovation, nor flexibility of pulse. By comparison, the collaborative intelligent, learning agent (a) acts and reacts and adapts to many more parameters in combination, and (b) is enabled, by its intelligent learned situated knowledge, to produce innovative but congruent material. This permits greater freedom on the part of the live (human) performer.

The way in which these criteria can be arranged is similar to the categorisation method described by Reffat & Gero (2000) in Computational Situated Learning in Design, when describing shape semantics:

“These regularities are arranged in terms of categories and since these categories reflect the relationships and dependencies among shape semantics based upon where they were

used they are called situational categories”

...except that in the collaborative agent model advocated in this paper, the relationships and dependencies are modelled on design grammars observed by the collaborator. The regularities referred to by Reffat & Gero (2000) are recurrences in geometric shape grammars, whereas in this model, regularities are observed in parameters defined by the designer during customisation, which therefore means that the parameters themselves differ vastly depending on the creative medium of interest and application. The subset of categories or parameters of greatest significance change over time and the dependencies also contribute to the grammatical knowledge learned by the collaborative agent as these, too, change throughout the course of the designing process.

It is obviously apparent that the design environment to which this concept is applied determines the mechanical or electronic receptors/ perception devices/data to which the computational model reacts. Hypothetically, in the case of the architectural design model in its final stages of rendering, for which the collaborator offers compatible textural alternatives, it might be driven through AutoCAD and computational application. The scenario of musical collaborator with live improviser, on the other hand, requires receptors capable of receiving rhythmic and pitch and other data about the design output as the connective interface with the computational application. This is achievable using a MIDI acoustic instrument or hyper-instrument or an instrument that produces the designers output electronically, e.g. a MIDI violin or a keyboard synthesiser or computerised music. The sophistication of the MIDI violin, for example, is such that its receptors, located in the fingerboard and bridge and/or body of the instrument, pick up information about the most critical parameters of the designers/performers musical output, e.g. pitch, rhythm, dynamic (loudness), attack, decay, tone quality, even which of the four strings is used to produce the tone. This information can be analysed according to the categories of customisation in the collaborative device immediately before it produces an adaptive or reactive response according to its learned knowledge about the designers grammar.

Some specific instances of selective criteria for collaborative response are modelled in Sections 5.1 (Scenario Model Intelligent Dynamic Design Collaborator for Musical Improvisation) and 5.2 (Selectivity and Categorisation Criteria to Classify Parameters, Subcategories and Interdependencies).

3.2 Reacting to Dynamic Design Input

A “reactive” response is one in which the collaborator provides a simple response in answer, reciprocation or reply to the material it receives. No computational transformation of context or material is required in this reaction. An example of this response is if the collaborator establishes that no change in dynamic or deviation from

its learned knowledge about the material is required before it provides a contextual response. This is the simplest kind of response the collaborator would return although, unlike the passive and unintelligent existing collaborative devices that only filter or modify the exact output of the designer, the collaborator still “creates” new material. The new material returned, however, is a “composition” or “design” constructed from criteria that are unmodified temporally, i.e. while the design environment is consistent or static in grammatical terms and according to the other parameters on which the model operates. As soon as a dynamic change occurs in the nature of the categories dominating the design process, the collaborator must instead react adaptively using knowledge of the designers creative process.

An example is the musical collaborative agent that observes a consistent pattern in the parameters of precedence. That is, if pitch, rhythm and dynamic are customised as criteria of dominant importance, and the designers output remains constant in its pitch language, rhythmic divisions, meter and pulse and the dynamic (loudness) is also constant, the collaborator reacts in a manner already familiar to it without having to further apply its intelligence about the designers creative practices. It still, however, generates innovative material. To simply replicate the designer would result in mimicry. The purpose, once again, of the collaborative tool, is to add to and complement the dominant designer and not to replicate his/her output, in a consistent style of creativity.

3.3 Providing an Adaptive Response to Input in Dynamic Situated Design

For a learning agent to be situated in the design process it should be able to learn the interrelationships among design knowledge and actions based on where they were used and be responsive to the changes that take place in the environment (Reffat & Gero, 1999).

It is in the dynamic design environment in which the collaborative model exercises its intelligence, utilising learned design behaviours from the principle designer. It uses this learned information provide an adaptive response to a changed design environment with which it is unfamiliar, i.e. the collaborator designs an appropriate response to a novel situation.

In a new or changed design situation, the collaborative tool combines its accrued knowledge, drawing on parameters of consistency while innovative in parameters of change. If, for example, the designer/improviser modulates musical key, the collaborative agent takes knowledge about pitch patterns and rhythmic patterns typical of the designer and transposes it into the new key. The collaborative agent may also have learned in its recent situated experience, that the designer when modulating to remote keys has the interdependent habit, or characteristic grammatical behaviour, of decreasing dynamic (loudness) intensity and, in this case, the collaborator combines the

dependent characteristics of these parameters in the formulation of its new material.

This scenario describes relatively simplistic act of transposition combined with innovation. The most challenging situation arises when the collaborator observes a completely novel set of parameters, i.e. when a disjunction occurs in the dynamic design environment. It is in this situation that its most sophisticated or intelligent computation takes place. The collaborative agent operates with the completely novel data set pertaining to immediately relevant constituent parameters and the characteristics of those parameters. It combines this material with processes of design learned from the design by its situatedness and generates innovatively on both levels. Only a situated agent with intelligent learning capability is able to respond to such a dynamic design environment.

3.4 Building a Predictive Behaviour in a Dynamic Situated Design

Each classification of response considered thus far, constitutes some variation of response to a set of conditions and learned creative methodologies based on the designers grammar. It follows that to increase the speed and human-like potential of the collaborator in a dynamic situated design, the collaborator could enhance its performance by "anticipating" designer actions using its learned knowledge about the designers inventive mannerisms.

This concept takes the collaborative intelligence one step further by utilising "expectation" or "anticipation" of the improvising or creating musician/designers future actions. Given that the collaborator learns historically about design practice, it is possible to implement this learned knowledge about the way in which the designer typically constructs change to predict a likely next step in the creative design. Like the adaptive response, this predictive response is only valid if checked against the currently extant conditions in the dynamic design environment. While not entirely different from the adaptive response in outcome, the temporal advantage in predicting or "pre-meditating" a possible solution to the collaborators creative task, is to facilitate a rapid response that can be computed during periods of design stasis, needing modification only to fit the dynamic parameters of the design at any given time.

The concern of this paper is not to consider analysis methods or data collection problems because there are already many commercially available and well-developed tools that are capable of many of the computational stages between the produced design (sound) emitted by the designer and the collaborative computer (that will be cited in section 5.). The conceptual Artificial Intelligence challenge in this conceptual model is to process the collected intelligence and learned knowledge and transform it into a classification of knowledge that may be operated upon creatively and provide feedback to the designer.

4 Interactivity between Designer and Collaborator

Although the principal characteristics of the collaborative agents intelligence are its ability to innovate a reactive response and its ability to create an adaptive response in a changing, dynamic design environment from which it has learned knowledge about the design process, there is also some degree of interactivity involved.

The interactivity occurs in three ways: firstly, the collaborative agent is reacting and adapting to a stimulus, namely the output by the principal designer. This is observed as an interaction on the part of the collaborator. Secondly, the material produced by the collaborator, in turn, will invoke reaction and action from the principal designer, thus forming some feed into the subsequent material generated by the designer. Thirdly, this reactive response to the collaborators material contributing to the output by the designer forms the trigger stimulus for further collaborative response. This builds a complex interrelation of the material created by both designer and collaborative that evolves in a close-knit outcome while concurrently undergoing evolution. While not constituting true interactivity, the composite result will be a complex union of relating reactive and adaptive responses, the sum of which can be inferred as interactive, especially when the composite design of both designer and collaborator is observed concurrently by a third party.

5 Scenario Model Intelligent Dynamic Design Collaborator for Musical Improvisation

5.1 Modelling Information Flow between Designer and Collaborator

Figure 1 illustrates paths for information flow between designer, collaborator and back to designer and audience. As input is classified, observed and concurrently behavioural knowledge is learned, whether the condition of the situated design environment is static or dynamic, and to what degree change has occurred, determines the reactive or adaptive process applied. Interaction occurs with the designer and the collaborator, both of whom "hear" (analyse) and respond to emergent innovation. Constantly, the collaborator generates predictive material verified against observed parameters.

Existing technologies provide solutions for many of the informational and linkage issues raised in this conceptual model. Some potential commercially available solutions are advocated here to address those technical interfaces that have already been addressed in existing but different ventures (proving technological validity):

Table 1: Existing Technologies for the Analysis Process

MIDI Output devices that convey complex digitised (binary) information about musical parameters include devices such as MIDI synthesizer keyboards, MIDI violin. Alternatively, if the designers tool is already computerized, such as in digital music production (composed/improvised on the computer), the task of digitisation is not required. It is necessary to digitise information about the design that would otherwise be an analogue sound signal, in order for the computational operations to be completed.

Chunking tools are those applications capable of dividing a continuous flow of information into musically logical and recognizable units upon which it is appropriate to apply isolated transformations. In musical language, a chunk is equivalent to a musical phrase at the semantic and phonetic level. In a phonetic sense, phrases are typically separated by small breaks or temporal pauses, separations in sound and semantically this analysis can be augmented with knowledge about typical grammatical progressions (e.g. cadences in tonal keys, etc.). A number of commercial software programs have an automated process for dividing music into small, processable phrase equivalents, e.g. Cool Edit 2000 a sound wave editing program. The program bases its “decisions” on the dynamic envelopes and pauses in the audio wave.

Tools that convert MIDI input into distinct musical parameters exist in the form of notation programs, e.g. Sibelius or Finale, that transform played-in music into score notation. The same capability holds potential to convert played-in music to category sets for computational intelligence.

Quantisation or discriminating between pitches and durations based on a customisable latitude of definition / scope in defining characteristics also occurs in notation programs. The degree of latitude or accuracy is typically programmed by the user, e.g. the measure of rhythmic flexibility or variance in pitch intonation permissible.

5.2 Selectivity and Categorisation Criteria to Classify Parameters, Sub-categories and Interdependencies

Table 2 demonstrates examples of criteria that might be used to categorise and differentiate between different characteristics of data input on which, in turn, observations about recurrent usage and localized, situated knowledge can be developed. The relative hierarchy of these

elements and constituent quantifiers will be transient or dynamic according to the situated design. Recognition of priority and quality/character form the basis of observations from which the collaborator learns.

Curry and Wiggins (1999) refer to the chunks or phrase units to which such categorisation is applied as the “Grouping Structure”.

“Grouping Structure is the segmentation of musical events into groups of similar or related events. These rules try to encapsulate the notion of “chunking” in which a listener groups certain events together whilst hearing the piece” (Curry & Wiggins, 1999, p.14-15).

Curry and Wiggins consider a similar task of dividing music into units or blocks that both make sense for the listener and for interpolation by computation. They describe A New Approach to Cooperative Performance: A Preliminary Experiment in which the relationship between high-level structure of a musical piece and a (human) performers expression is investigated in order to model an expressive computer performance. While their goal is different to the collaborative agents in this paper, their criteria for dividing information and categorising it are relevant to the concerns of this paper in deducing empirical data from subjective human performance for the purposes of computational interpretation.

Once categories are classified for analysis, the high-level structures can be represented with logical formulae allowing the expression of complex relationships between different musical categories in divisions equating to “descriptions”.

In Music Representation - Between the Musician and the Computer (Smaill, Wiggins & Miranda, 1993), hierarchical relationships are expressed as definitions based on logical properties of constituent notes (pitches) n_1, n_2 , and constituents or characteristics related to these notes C_1, C_2 . The frequency or pitch of n is an absolute value of the pitch parameter as described in Table 2. above and C refers to the values attributed to category (and further sub-class) of pitch in the same table. On this basis, a new constituent or description of the (phrase) unit based on the logical properties of n and C is:

$$D1 = \{x : P(x)\} \quad (1)$$

where P is a logical formula, gives a constituent that selects those values of n (pitch) and C (pitch category) that make P true. Combinations can be explicitly expressed:

$$D2 = \{n_1, n_5, C_2, C_3\} \quad (2)$$

The expression of combinations distinguishes interdependencies. This is clearly much simpler than the computational reality for which the number of n variables might be in the vicinity of 200 or more and the number of C possible variables would depend on the parameter being

described and the specificity of customisation of the program. Given that for every phrase unit there are multiple parameters and, for each parameter, qualifying categories and sub-classes, the actual representation would be considerably more complex.

It is beyond the scope of this paper to classify subjective, qualitative genres in musical style that affect design grammar. The reader is directed to a further paper by Wiggins: Towards a more precise characterization of creativity in AI, (Wiggins, 2001); in which he formulates detailed and logical descriptions of different historic musical genres and their typical characteristics in mathematical syntax. This is necessary for the performance of any computation.

6 Conclusion

Existing computational agents or assistants used to create musical design either innovate new designs alone based on structures that have been learned from human creativity or other methodologies for generating new material.

Existing accompanimental assistants or collaborative tools passively reinterpret human performers or act on pre-programmed structures. To innovate new material from learned knowledge in an evolving dynamic design environment, in real time and in response to a designer, requires a different level of intelligence that is both sensory and observatory.

The situatedness of this concept model is fundamental to determining the type of response this model provides, depending on the appropriateness of the type of response it gives. This model undertakes a reactive or adaptive response, in combination with prediction and constant comparison with the dynamic, current conditions of its design situation through observation.

Amongst the characteristics that change in a dynamic design environment are the constitution and hierarchy of parameters governing design creativity, in addition to the character or nature of these parameters. Prediction and evolving methodology for improvisation builds intelligently on learned knowledge about the design situation and the designers creative process.

The human designer remains the stimulus for collaboration and creative practice and the collaborators role is inseparable, in time and situation, from the situated design and situated learning that emanates from the principal designer.

This concept has the potential to improve productivity, innovation, invention and true, human-like collaboration and to provide intelligent, dynamic responsiveness in real time.

Acknowledgements

The author would like to acknowledge the extensive research in the Key Centre of Design Computing and Cog-

nition in the Faculty of Architecture at the University of Sydney, especially by Professor John Gero. The author looks forward to future work with Greg Smith in the Faculty of Architecture to develop the mathematical substantiation of the model discussed herein.

References

- Bryson, Smaill, A. & Wiggins, G.: 1992, *The Reactive Accompanist: Applying Subsumption Architecture to Software Design*.
- Curry, B. & Wiggins G.: 1999, *A New Approach to Cooperative Performance: A Preliminary Experiment*, *International Journal of Computing Anticipatory Systems*.
- Gedenryd, H.: 1998, *How Designers Work*, Ph.D Thesis, Lund University, Lund, Sweden.
- Gero, J. S. and Fujii, H.: 2000, *A computational framework for concept formation in a situated design agent*, *Knowledge-Based Systems* 13(6), pp. 361-368.
- Harris, Smaill, A. & Wiggins, G.: 1993, *Representing Music Symbolically*, IX CIM, Genova.
- Johnson-Laird, Philip N.: 1991, *Rhythm and Meter: A Theory at the Computational Level*, *Psychomusicology* 10, 88-106.
- Padgham, L.: 2001, *Smart Software Agents with Personality and Intelligence*, in Nolch, G. (ed.) *Australasian Science*, Vol. 22(10), Control Publications Pty. Ltd., Australia, pp. 33-36.
- Papadopoulos & Wiggins, G.: 1999, *AI Methods for Algorithmic Composition: A Survey, A Critical View, and Future Prospects*, *Proceedings of the AISB'99 Symposium on Musical Creativity*, AISB.
- Papadopoulos & Wiggins, G.: 1998, *A Genetic Algorithm for the Generation of Jazz Melodies*, *STeP'98*, Jyväskylä, Finland.
- Park, S-H. and Gero, J. S.: 2000, *Categorisation of shapes using shape features*, in Gero, J. S. (ed.), *Artificial Intelligence in Design00*, Kluwer, Dordrecht, pp.203-223.
- Pearce & Wiggins, G.: 2001, *Towards A Framework for the Evaluation of Machine Compositions*, in Wiggins, G. (ed.) *Proceedings of the AISB'01 Symposium on AI and Creativity in Arts and Science*, AISB.
- Reffat, R. and Gero, J. S.: 2000, *Computational situated learning in design*, in Gero, J. S. (ed.), *Artificial Intelligence in Design00*, Kluwer, Dordrecht, pp. 589-610.
- Reffat, R. and Gero, J. S.: 1999, *Situatedness: A new dimension for learning systems in design*, in A.

- Brown, M. Knight and P. Berridge (ed.s), *Architectural Computing from Turing to 2000*, eCAADe, University of Liverpool, U.K., pp. 252-261.
- Robertson, J., de Quincey, R., Stapleford, T., & Wiggins, G.: 1998, *Real-Time Music Generation for a Virtual Environment*, ECAI98 workshop on AI/Alife and Entertainment, Brighton, England.
- Smaill, A., Wiggins, G. & Harris: 1993, *Hierarchical Music Representation for Composition and Analysis*, *Computing and the Humanities Journal*.
- Smaill, A., Wiggins, G. & Miranda: 1993, *Music Representation - between the musician and the computer*, World Conference on AI and Education workshop on Music Education, Edinburgh.
- Wiggins, G., Papadopoulos, Phon-Amnuaisuk & Tuson: 1999, *Evolutionary Methods for Musical Composition*, *International Journal of Computing Anticipatory Systems*.
- Wiggins, G. & Smaill, A.: 2000, *Musical Knowledge: what can Artificial Intelligence bring to the musician?* (Ch.) in Miranda (ed.) *Readings in Music and Artificial Intelligence*, Harwood Academic Publishers.
- Wiggins, G.: 2001, *Towards a more precise characterisation of Creativity, AI*, Proceedings of the ICCBR 2001 Workshop on Creative Systems, Vancouver, British Columbia.
- Wiggins, G., Miranda, Smaill, A. & Harris: 1993, *Surveying Musical Representation Systems: A Framework for Evaluation*, *Computer Music Journal*.
- Wiggins, G., Harris, M. & Smaill, A.: 1989, *Representing Music for Analysis and Composition*, EWAIM89, Genova.
- HREF1: <http://www.york.cuny.edu/seitz/JPS2001paper.htm>, last visited 1/11/01
- HREF2: <http://citeseer.nj.nec.com/context/208274/0>, last visited 1/11/01
- HREF3: <http://citeseer.nj.nec.com/ramalho94simulating.html>, last visited 1/11/01

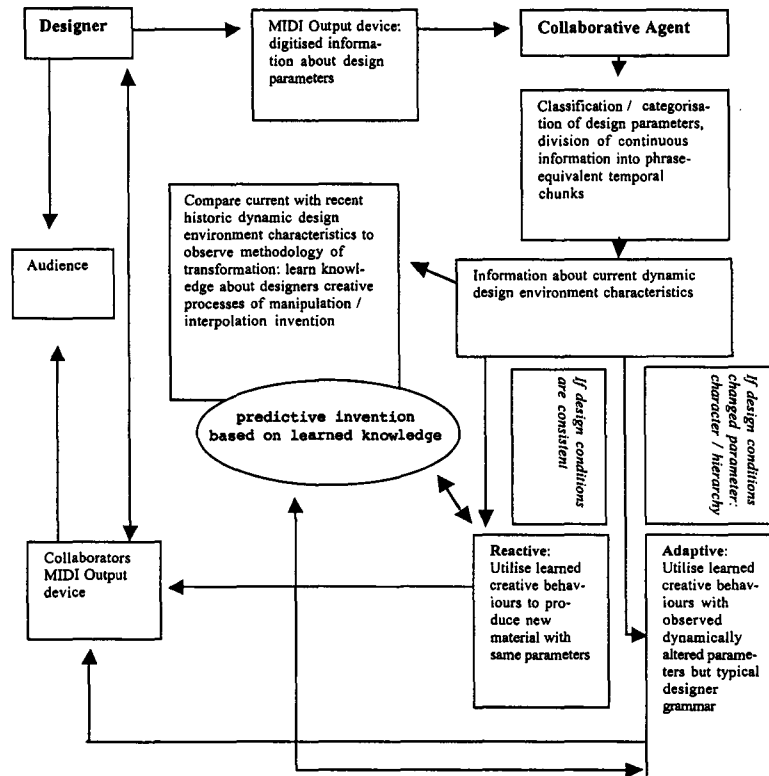


Figure 1: Information flow between the designer, collaborator and audience; processed according to dynamic stasis/change in the situated design environment.

Table 2: Categorisation Criteria Applied to Chunks/Phrase Units

Parameters	Categories	Sub-classes	Scope/Limiters
Pitch/Modality	Tonal, chromatic, modal, microtonal	Different modes, key relations, micro-intervals	Minimum deviation (Cents) differentiated & maximum differentiation permitted
Meter	Simple/compound; additive; regular/irregular	Duple, triple, quadruple; simple/complex divisions/irrational	Maximum deviation subdivision unit
Pulse/Tempo	Beats, speeds, syncopation, cross-rhythmic subdivisions	Emphases within the bar/measure, accentuated pulses	Variance from strict tempo, minimum and maximum speeds, variation in accuracy
Rhythmic Duration	Different units of duration interdependent on meter	Whole units, irrational fractions of units inter-dependent on meter	Durational variance permissible relative to unit
Attack	Velocity, Envelope, Size/degree, nuance	Percussive, subtle, inarticulate, joined to previous pitch	Degree of distinction between sub-classes
Dynamic Loudness	pppp--fff	Intermediate values	Hertz differentiation between increments
Timbre	Tonal Quality (depends on sound medium)	Refined subsets within categories, related tonal qualities	Sensitivity of perceptual device
Registration/Input	Different strings or registers of input device	Regions high and low relative to category	Sensitivity of perceptual device

II – Patterns and Structure in Creative Systems

Giving Colour to Images

Penousal Machado^{1,2}; André Dias²; Nuno Duarte²; Amílcar Cardoso²

¹ Instituto Superior de Engenharia de Coimbra; Quinta da Nora, 3030 Coimbra, Portugal;

² CISUC – Center for Informatics and Systems, Univ. Coimbra, 3030 Coimbra, Portugal;
machado@dei.uc.pt; adias@student.dei.uc.pt; nduarte@student.dei.uc.pt; amilcar@dei.uc.pt;

Abstract

This paper is about the colouring of greyscale images. More specifically, we address the problem of learning to colour greyscale images from a set of examples of true colour ones. We employ Genetic Programming to evolve computer programs that take as input the Lightness channel of the training images and output the Hue channel. The best programs evolved can then be used to give colour to greyscale images. Due to the computational complexity of the learning task, we use a genome compiler system, GenCo, specially suited to image processing tasks.

1 Introduction

The work presented here is part of a wider research project, NEvAr, whose aim is to build a constructed artist (i.e. a program that generates artworks autonomously).

NEvAr is an Evolutionary Art Tool inspired on the work of K. Sims (1991). It relies on Genetic Programming (GP) to evolve populations of images, based on aesthetic principles. Fitness assignment plays, like in most Evolutionary Computation systems, a key role since it guides the evolutionary process.

NEvAr can be used as an Interactive Evolution tool. In this mode of execution the user supplies the fitness values to the evolved images. It can also be used as a fully autonomous system. In this case fitness is assigned through an explicit fitness function, which takes into consideration several complexity estimates of the images (Machado, 2002).

However, the fitness assignment procedure only takes into account the lightness information of the images, discarding the hue and saturation information. Therefore, in this mode of execution, we are limited to greyscale images. A full description of NEvAr and of the automatic fitness assignment can be found in (Machado, 2002).

There are good theoretical and artistic reasons to deem colour less important than lightness. The development of the colouring procedures of systems like AARON (Cohen, 1995) is, to some extent, based on this notion. However, this collides, at least apparently, with the importance given to colour by some of the most prominent

painters (e.g. (Kandinsky, 1991)). Moreover, NEvAr's limitation to greyscale images, in its autonomous version, was frustrating, to say the least. In this paper we address the problem of giving colour to greyscale images.

An analysis of the role of colour and the way colour is assigned, particularly in abstract art, leads to the conclusion that artists (certainly not all, but at least a significant proportion) usually work with a limited colour palette, and that the spatial relation between colours usually follows some rules. This is consistent with the view that each artist constructs its own artistic language, which complies with an implicit grammar. It is also consistent with the approach used in AARON to colour its drawings (Cohen, 1995; Cohen, 1999).

The idea of creating a program to give colour to the greyscale images created by NEvAr emerged naturally. Unfortunately this poses several problems. Our system is based on a non symbolic approach and produces bitmap images, hence there is no clear definition of closed forms, shapes, etc.

Therefore, although we could define a palette to work with, assigning the colours of that palette to specific forms would be difficult since we have no forms to begin with. Even assuming that the forms could be properly identified by some sort of pre-processing method, assigning the right colour to each shape and keeping a proper spatial relation among colours would still be a problem, due to the unstructured nature of the output.

Additionally, the creation of a colouring system, by itself, doesn't appear to be an easy task, involving the

choice of an adequate set of palettes, establishing a consistent colouring grammar, etc.

Taking these facts into consideration, and also the fact that the generality of this type of approach would be limited, we decided to abandon this idea. Instead, we are trying to create a system that learns to colour images from a set of training ones.

This approach has, potentially, several advantages over a built-in colouring procedure, namely: we don't need to code by hand a set of colouring rules; the results of the system are less predictable; we can use paintings made by well-known artists as training set, hence learning to colour images according to their style.

Additionally, it's also an indirect way of testing if the colouring procedures followed by some artists can be formally expressed.

The paper is structured as follows: In the next section we describe our current approach to colouring images; In Section 3, we present some experimental results attained by this approach and make a brief analysis; finally, in section 4, we will draw some conclusions and present our ideas for future research.

2 Our Approach

GP is one of the most recent Evolutionary Computation techniques. Its goal is to evolve populations of computer programs, which improve automatically as evolution progresses (Banzhaf 1998).

Due to the outstanding influence of the work of Koza (1992) it is common, within the Machine Learning community, to associate the term GP to the evolution of tree structures. In this paper we follow this "classical" definition. Therefore, when we talk about GP we are talking about the evolution of tree structures, which are built from a set of functions (f-set) and terminals (t-set). The internal nodes of the tree are members of the f-set, and the leafs are members of the t-set.

In our approach we use GP to evolve populations of programs that give colour to greyscale images. We start by selecting a true-colour training image (or set of images), which is split in its Lightness, Hue and Saturation channels (see Fig. 1).

The evolved programs take the greyscale image corresponding to the Lightness channel as input, and output a

greyscale image. The output is compared with the Hue channel of the training image, the closer the output is to the desired one, the higher the fitness. The same procedure can be applied using the Saturation channel, to evolve programs that generate the saturation information.



Figure 1: The original image, and the corresponding Lightness, Hue and Saturation channels.

As evolution progresses, the quality of the individuals increases and, eventually, we find programs which generate colourings very close to the original ones.

It is important to notice, however, that this is not enough – the idea is to use these programs to give colour to different images. The fact that a program produces an output that exactly matches the training one does not guarantee that it will produce an interesting colouring on different images. In other words, the evolved programs must generalise well. To promote their generalising capabilities, we took some precautions in the selection of the function set and also in the construction of the fitness function. In the remainder of this section, we describe the options taken and give justification for these options.

2.1 Implementation details

A first word goes to how the output of each program is calculated: given a particular individual, it is run for each of the pixels belonging to the training image (or images). Therefore, assuming a training image of 100*100 pixels, each individual must be run 10000 times. Each execution of an individual implies the transversal of its tree and calling, for each node the corresponding function.

When we take into account that the individuals can easily reach sizes of several thousand nodes, and that GP populations usually contain several hundred individuals, the

conclusion that the execution step is of considerable computational weight clearly follows. In order to minimize this problem we implemented our system with GenCo, a Genome Compiler system specially suited for image processing tasks (Machado, 2001).

A Genome Compiler is a GP system that makes online compilation of the evolved programs. In situations like the one previously described, i.e. in which each individual must be executed several times, this type of system can provide significant speed improvements, since each individual is compiled once and the resulting machine code executed several times (10000 considering a training image of 100*100 pixels). In this scenario, genome compiler systems are, typically, 50 to 100 times faster than standard C based GP implementations (Fukunaga 1998; Nordin 1994; Nordin 1995; Machado 2001).

Next we describe the design options made in the selection of the function and terminal set, and the reasons that justified them.

2.1.1 Function and Terminal Set

Our problem has some similarities with the symbolic regression of functions or images (Nordin 1995; Koza 1992). There are, however, some important differences. In a symbolic regression task, one would usually resort to a function set composed by the arithmetic operations and the *if* statement, and use as terminal set the variables X, Y. This type of set-up unavoidably results in programs whose output depends exclusively on the coordinates of the pixel being calculated.

In an attempt to solve this problem, we added to the terminal set the lightness value of the pixel being evaluated, thus giving more information to the program. This allows the output to vary in accordance to changes on the lightness channel. Additionally, we also added the lightness values of the adjacent pixels, so that the programs have access to the surrounding context of each pixel.

In what concerns the function set, we decided to keep the "traditional" one. The reason for this choice is threefold: it was deemed sufficient for a first approach; the inclusion of more complex functions (e.g. for-next loops, or high level constructs) would severely increase the computational weight of the evaluation step; the inclusion of this type of functions doesn't necessarily benefit the evolutionary process, in fact, the opposite can, and frequently happens (see, e.g., (Banzhaf, 1998)).

Taking all this factors in consideration we used the following function and terminal sets:

- F-set = {+, -, *, %, if}, where % stands for the protected division operator, and if for the *if-less-then-else* statement (Koza, 1992)
- T-set = {X, Y, A..I}, where X, Y are variables corresponding to the coordinates of the pixel, and A..I are the lightness values of the pixel being calculated and of the surrounding ones.

The results achieved were very disappointing, basically because the behaviour of programs continued to be determined, almost exclusively, by the values of the X and Y variables. This is clearly undesirable, since it means that we are not evolving programs that give colour to images according to their lightness channel, we are merely evolving a function that, when applied over an interval of X, Y values, generates the hue channel of the training image. Thus, we are only memorizing the training instance(s).

Taking the X, Y variables from the terminal set resulted in having poor evolution, and convergence to trivial and uninteresting colourings (e.g. having as a result the predominant colour of the training image, or always assigning to a specific lightness value the same hue or saturation).

It was clear that the variables X, Y were not producing any desirable impact on the programs. Therefore, we decided to delete them from the terminal set. It was also clear that without these variables the programs hadn't enough information to determine the appropriate colour for each pixel, since the programs only have a local view of the lightness channel (the pixel being evaluated and the nine surrounding ones). To compensate this lack of information, we introduced a new function, *get*($\Delta x, \Delta y$), whose behaviour can be described as follows: assuming that the current pixel has the coordinates a, b, it returns the lightness value of the pixel situated on ($a + \Delta x, b + \Delta y$).

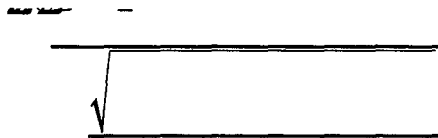
Since the *get* function provides a way to access the lightness values of the surrounding pixels, there was no reason to make these values as part of the terminal set. Accordingly, our function and terminal sets became:

- F-set = {+, -, *, %, if, get}
- T-set = {E}, where E stands for the lightness value of the pixel being evaluated.

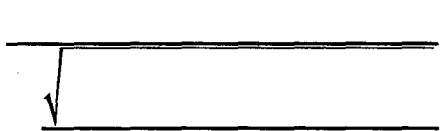
This set-up proved to be adequate, allowing the evolution of suitable colouring programs with good generalization capabilities, as will be shown in section 3. In the following section we focus on the used fitness functions, and on the grounds for using them.

2.1.2 Fitness Functions

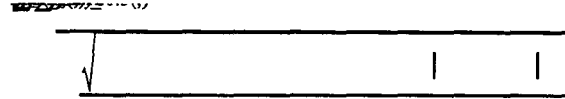
We start by describing the fitness function used when we are evolving the saturation channel of an image from the lightness one. In this situation we use the root mean square error between the desired output and the real one to assign fitness, i.e. considering that I holds the desired saturation values and that P is the output of a program the fitness is given by the following formula:

$$\sqrt{\frac{1}{n} \sum_{i=1}^n (I_i - P_i)^2} \quad (1)$$


When we are trying to evolve the Hue information, this formula must be slightly altered due to the circular nature of the Hue channel. Assuming that the images take values in the $[0, 1]$ interval, the above formula yields a maximum distance when $I(x,y) = 1$ and $P(x,y) = 0$ (or vice-versa). However, in this situation the difference in hue would be quite small and probably unnoticeable to a human observer. Therefore, we use the following formula:

$$\min_{\theta} |I - P + \theta| \quad (2)$$


where I holds the desired Hue values, P the program's output, and \min_{θ} returns the minimum angle distance between the two values. This fitness function can be further improved if we take into consideration that the perceived difference in hue depends on the lightness of the pixel (e.g. if the pixel as lightness equal to zero it will always be black, no matter what hue we assign to it; even when the lightness is only close to zero, giving to that pixel a hue different than the desired one will hardly be noticeable to a human viewer). Taking this into account, and considering that L holds the lightness information we obtain the following formula:

$$\sqrt{\frac{1}{n} \sum_{i=1}^n (I_i - P_i)^2 \cdot L_i} \quad (3)$$


The tests conducted using formulas 2 and 3 as fitness functions yield deceptively good results. In the early steps of evolution fitness increases steadily and swiftly, giving the impression that an adequate colouring program will be easily found. However, after a few hundred generations, the improvements in fitness drop suddenly and evolution seems to halt. This wouldn't be a cause for concern if the evolved programs solved the problem at hand satisfactorily. To some extent they do, since they usually have high fitness values, relatively close to the maximum attainable fitness. The problem is that a qualitative analysis of the results reveals that the colourings are usually uninteresting from an aesthetic perspective – typically, only a small subset of the original colours is used, resulting in the loss of nuances that make the original colouring work. This situation becomes more severe when the training images have a large area filled with a particular hue value, and other areas filled with hue values close to that one.

To better explain the problem we will use an example: consider, for instance, a landscape painting mostly filled with green (for the grass and trees) and with a blue sky (green and blue are relatively close in the colour spectrum); a program that outputs the hue corresponding to the green colour will have a relatively good fitness, since it is not penalised in the dominant green area of the image and only suffers a little penalisation on the blue area. To make things even worse, such a program is easy to evolve, so it will be found in a small number of generations. Changing it by mutation or crossover will tend to decrease its fitness, meaning that the fittest descendents will tend to be similar to it. Therefore, in a few generations the population will be dominated by similar programs, which will begin to increase in their size to protect themselves from destructive crossover and mutation; soon the increase in size becomes exponential and evolution becomes impossible (the exponential growth of program size is usually called bloat problem; a more detailed explanation of why bloat occurs can be found in (Banzhaf, 1997)).

In an attempt to force our system to evolve more interesting colourings, covering a wider colour spectrum, we decided to add another factor to our fitness function. A description of the procedure used to calculate that factor follows.

We start by taking the I image (that holds the desired hue values) and decrease its colour depth, using the optimised median cut algorithm, obtaining an image I' composed by only 16 different hue values ($v_1 \dots v_{16}$). Then we count how many pixels exist of each different hue and store the pixel count values in an array, $A_{I'}$. For each pixel of image P , which holds the output of the program, we determine the closest hue value, v_i , and the distance Δv between the pixel value and v_i . We add to $A_{P[i]}$ the value $1 - \Delta v$, thus if the pixel's hue matches exactly one of the sixteen hues present in the training image we add one, when it doesn't match exactly we add slightly less. After performing this procedure for all the pixels of the output image we compute the following formula:

$$\frac{\sum_{i=1}^{16} |A_{I'}[v_i] - A_P[v_i]|}{16} \quad (4)$$

Thus, we are basically comparing the number of pixels of each hue value of the output and target image. Returning to our previous example of the green and blue landscape, if the output image has the same amount of blue pixels and green pixels than the original (and also assuming that it is the exact blue and green tone), formula 4 will give a value close to one. However, if the output image is dominated by the green colour there will be a huge discrepancy between the amount of green and blue of the two images, and therefore H_b will be close to zero. Notice that, in what H_b is concerned, the placement of the colours has no real influence on the resulting value. To further force a wider coverage of the colour spectrum, we also used the following modification to this formula:

$$\frac{\sum_{i=1}^{16} |A_{I'}[v_i] - A_P[v_i]|}{\sum_{i=1}^{16} A_{I'}[v_i]} \quad (5)$$

which ensures that all sixteen different hues have the same weight in the calculation. Resorting again to our example, and considering that we have a small yellow area (for the sun), when we use formula 5 having the correct number of yellow pixels is as important as having the correct number of blue or green ones. In the next sec-

tion we will present some of the experimental results achieved.

3 Experimental Results

To test our approach we conducted a series of experiments. As training images, we used some of the early works of Wassily Kandinsky. The images were reduced to the size of 96*96 pixels in order to allow a faster evolution. Although our system may use several training images at the same time, we haven't taken advantage of this possibility so far. The results presented in this section concern the evolution of programs that take the lightness channel of an image as input and give the hue channel as output. As fitness we used $H_a + H_b$.

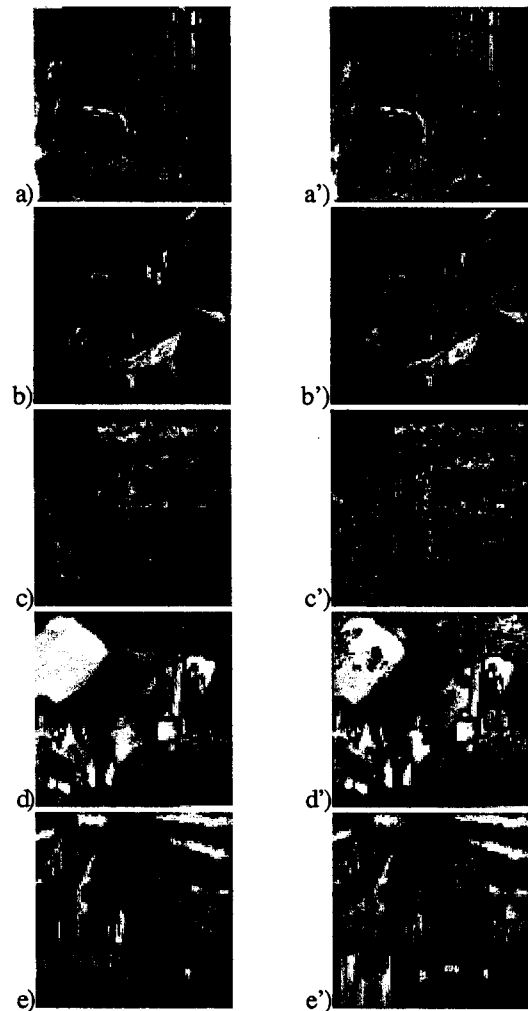


Figure 2: On the left column the original images, on right the images resulting from the application of the best individual of each run to the lightness channel of the training image.

In Fig. 2 we present some of the training images, and the images generated by the evolved programs¹.

It is clear from the results presented that we can achieve images very close to the original ones. However, what is really important to our goal is accessing how well do the evolved programs perform when applied to images not involved in their training, i.e. their generalisation capabilities. In Fig. 3 we present some results achieved with this mode of operation.

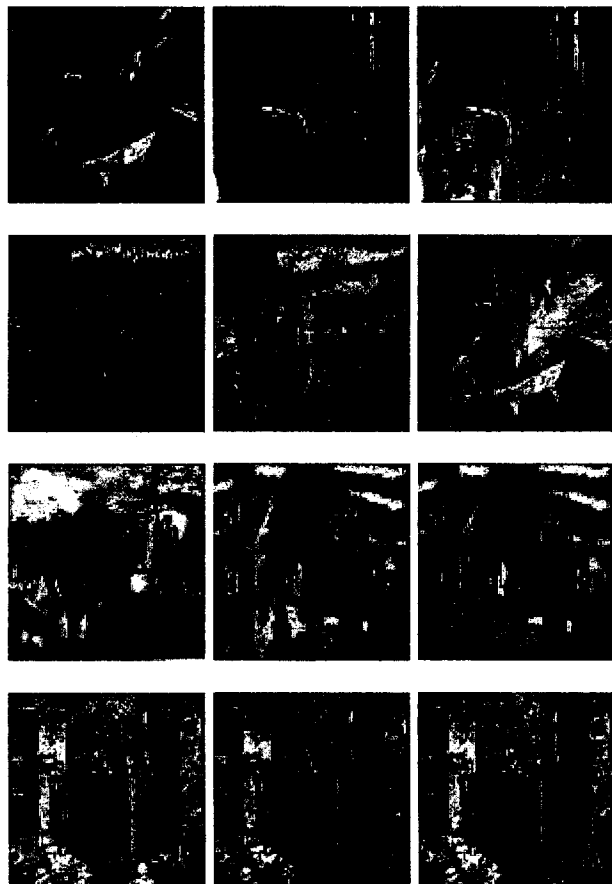


Figure 3: The images on the first column result from the application of the program that generated the image a' from Fig. 2; on the second column, images resulting from the application of b' ; on the third column images resulting from the application of c' .

We consider these results to be extremely promising. Some of the colourings presented in Fig. 3 are quite close to the original ones and, additionally, in some cases, al-

¹ A colour version of the paper can be found in:
<http://www.dei.uc.pt/~machado/research/research.htm>

though significantly different, they are still interesting colourings. In order to allow a more equitable assessment of the results, we don't present the original image corresponding to the fourth row of Fig. 3.

4 Conclusions and further work

In this paper we presented our ongoing research whose goal is to learn to colour greyscale images from a set of training instances. This effort is part of a wider research project that aims at building a fully autonomous constructed artist.

The results achieved so far concern, mostly, the evolution of programs that generate the hue channel from the lightness one. The experiments performed on the evolution of programs to generate the saturation channel, seem to indicate that this task is quite simpler. Nevertheless, the replacement of the original saturation channel by one generated through a program will unavoidably imply the introduction of further noise. At the time of writing we can't access exactly how much this will affect the quality of the generated colourings (although we foresee little impact).

There is also the need to conduct additional experiments, specially to use several training images simultaneously, which will hopefully increase the generalization capabilities of the evolved programs. Another aspect that can be improved is the fitness function, we are currently altering it so that it also takes into account the vicinity relations between areas of colour, once again the idea is to promote the evolution of programs that assign colour based in more general concepts.

A final word goes to other types of application of the proposed techniques, for instance, the colouring of black and white movies. Assuming that one of the frames is coloured (either by hand or by some other method) it should be possible to evolve a program that gives correct colours to the surrounding frames.

Acknowledgements

This research project was approved by FCT (Fundação para a Ciência e Tecnologia) and POSI (Programa Operacional "Sociedade da Informação"), and is partially funded by FEDER, project n. POSI/34756/SRI/2000.

We would also like to thank the blind reviewers of this paper, which provided precious remarks not only for the paper in question but also for future research.

References

- Banzhaf, W., Nordin, P. and Francone, F. D. Why introns in genetic programming grow exponentially, *Workshop on Exploring Non-coding Segments and Genetics-based Encodings*, ICGA, East Lansing, MI, USA, 1997.
- Banzhaf, W., Nordin, P., Keller, E. and Francone, F. D. *Genetic Programming – An Introduction*, Morgan Kaufman, 1998.
- Cohen, H., The Further Exploits of AARON, Painter, SHR, *Constructions of the Mind*, Vol. 4, Issue 2, 1995.
- Cohen, H., Colouring Without Seeing: a Problem in Machine Creativity, *AISB Quarterly*, 102:26-35, 1999.
- Fukunaga, A. Stechert, A. Mutz, D. A Genome Compiler for High Performance Genetic Programming, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pp. 86-94, Morgan Kaufmann, 1998.
- Kandinsky, W., *On the Spiritual in Art*, republished by Dover publications in 1997, 1911.
- Koza, J. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, 1992.
- Machado, P., Dias, A., Cardoso, A., GenCo – A project Report. *Proceedings of the Third International Symposium on Artificial Intelligence and Adaptive Systems (ISAS'2001)*, La Havana, Cuba, 2001.
- Machado, P., Cardoso, A., All the truth about NEvAr. *Applied Intelligence, Special issue on Creative Systems*, Bentley, P. Corne, D. (eds), Vol. 16, Nr. 2, pp. 101-119, Kluwer Academic Publishers, 2002.
- Nordin, P. A compiling genetic programming system that directly manipulates the machine-code. *Advances in Genetic Programming*, Kenneth E (Ed.), pp 311-331, MIT Press, 1994.
- Nordin, P., Banzhaf, W. Complexity Compression and Evolution, *Genetic Algorithms: Proceedings of the Sixth International Conference*, ICGA95, pp. 310-317, Morgan Kaufmann, 1995.
- Sims, K., Artificial Evolution for Computer Graphics, *ACM Computer Graphics*, Vol. 25, pp. 319-328, Addison-Wesley: Boston, MA, 1991.

Exact and Approximate Distributed Matching for Musical Melodic Recognition

Costas S. Iliopoulos; Masahiro Kurokawa
Algorithm Design Group, Dept of Computer Science
King's College London, Strand, England and
School of Computing, Curtin University of Technology, Perth, Australia
Email: {csi, kurokawa}@dcs.kcl.ac.uk

Abstract

Here, we present fast and practical algorithms for musical melodic recognition. Music analysts are often concerned with finding occurrences of patterns (motifs), or repetitions of the same pattern, possibly with variations, in a score. The pattern can be distributed horizontally in a score, either in one voice or across several other voices, and the pattern occurrence can be exact or with errors. Modifying the Shift-Or algorithm and considering the pitch and rhythm, exact and approximate distributed matching problems are solved in linear time with few false matches.

1 Introduction

The main motivation for studying the family of distributed matching problems is their application to computer assisted music analysis. Music analysts are interested in finding occurrences of patterns (*motifs*), repetitions of the patterns, possibly with variations (*errors*), in musical scores. A score can be viewed (at some level) as a string. The alphabet can be the set of notes in simple cases or the GRIP representation of Cambouropoulos (1996) in more general cases. Using this approach, the solution of several computational music problems can be done by means of pattern matching algorithms. These problems are extensions of the ordinary string matching problem: the difference is that the pattern or the text (or both) can be multiple strings called *plural strings*.

The primary goal in this paper is to design and implement bit-wise algorithms for several variants of the distributed matching problem: given a set of sequences of notes (one for each voice) and a pattern, find whether the pattern occurs distributed horizontally, either in one voice or across several other voices; the pattern occurrence can be exact or with errors.

Here we consider distributed problems related to exact and approximate string matching: exact distributed string matching and approximate distributed string matching. Formally, given s texts $T^i = t_1^i \dots t_n^i, i \in \{1 \dots s\}$ of equal length n over the alphabet Σ and a pattern $P = p_1 \dots p_m$ of length $m \leq n$ over the same alphabet Σ , the problem of the exact distributed string matching is to find all occurrences of the pattern P in the texts $T^i, i \in \{1 \dots s\}$, such that the various parts of the pattern P can be located in consecutive positions of different texts, i.e. find all positions $r \in \{1 \dots, n - m + 1\}$, such that for each

$j \in \{1 \dots m\}$, there exists an integer $l_j \in \{1 \dots s\}$ such that $p_j = t_{r+l_j-1}^{l_j}$. In the approximate distributed string matching problems we search for all occurrences of the pattern in the text with at most k errors. In this paper we consider the Levenshtein distance (replacement, insertion, deletion).

A survey of computational tasks arising in musical analysis can be found in Crawford, Iliopoulos, and Raman (1998). Distributed pattern matching problems were introduced in Holub, Iliopoulos, Melichar, and Mouchard (2001); they presented a finite automata for a singular/plural text vs a singular/plural pattern. Meredith, Wiggins, and Lemstöm (2001) presented new practical algorithms for the distributed pattern matching problem. Cole, Hariharan, and Indyk (1999) presented asymptotically fast $O(n \log n)$ algorithms for a variant of the distributed matching problem. Cole, Iliopoulos, and Rytter (2001) presented asymptotically fast algorithms for δ matching. The latter two algorithms require convolutions that are impractical to implement.

A new thread of practice-oriented results exploited the hardware word-level parallelism of bit-vector operations. Baeza-Yates and Gonnet (1992) presented an $O(nm/w)$ algorithm for the exact matching case and an $O(nm \log k/w)$ algorithm for the k -mismatches problem, where w is the number of bits in a machine word, n is the length of the text, and m is the length of the pattern. Wu and Mänter (1992) showed an $O(nkm/w)$ algorithm for the k -differences problem. Furthermore, Wright (1994) presented an $O(n \log |\Sigma| m/w)$ bit-vector style algorithm, where $|\Sigma|$ is the size of alphabet for the pattern. Wu, Mänter, and Myers (1996) developed a particularly practical realization of the 4-Russians approach introduced by Masek and Paterson (1980). Most recently, Baeza-Yates

and Navarro (1996) have shown a $O(nkm/w)$ variation on the Wu-Manber algorithm, implying $O(n)$ performance when $mk = O(w)$.

The most common variant of the approximate string matching problem is to find substrings that match the pattern with at most k -differences. The first algorithm addressing exactly this problem is attributable to Sellers' (1980). Sellers' algorithm requires $O(mn)$ time, where m is the length of the query, and n is the length of the text. Subsequently, this was refined to $O(kn)$ expected time (Ukkonen, 1985), then to $O(kn)$ worst-case time, first with $O(n)$ space (Landau and Vishkin, 1998), and later with $O(m^2)$ space (Galil and Park, 1990).

Here we employed word-level parallelism into algorithms and speeded up the solutions of the above computational problems. In particular we present a $O(n\lceil m/w \rceil)$ algorithm for exact distributed matching, where n is the length of the text, m is the length of the pattern, and w is the length of the computer word.

This paper is organized as follows. In the next section we describe the basic definitions and background required. In section 3, we describe the Shift-Or algorithm, and in section 4, we modify input data. We present algorithms for exact distributed matching problem in section 5, and for approximate distributed matching problem in section 6. We implement them in section 7, and we describe the asymptotic time complexity in section 8. Finally in section 9 we present our conclusions.

2 Background and basic string definitions

A *string* is a sequence of zero or more symbols from an alphabet Σ ; the *no symbol*, that is the string with zero symbols, is denoted by ϵ . The set of all strings over the alphabet Σ is denoted by Σ^* . A string x of length n is represented by $x_1 \dots x_n$, where $x_i \in \Sigma$ for $1 \leq i \leq n$. A string w is a substring of x if $x = uwv$ for $u, v \in \Sigma^*$; we equivalently say that the string w occurs at position $|u| + 1$ of the string x . The position $|u| + 1$ is said to be the *starting position* of w in x and the position $|u| + |w|$ the *ending position* of w in x . A string w is a *prefix* of x if $x = wu$ for $u \in \Sigma^*$. Similarly, w is a *suffix* of x if $x = uw$ for $u \in \Sigma^*$. The string xy is a *concatenation* of two strings x and y . The concatenation of k copies of x is denoted by x^k .

Consider two sequences $\tau = \tau_1 \tau_2 \dots \tau_r$ and $\rho = \rho_1 \rho_2 \dots \rho_r$ with $\tau_i, \rho_i \in \Sigma \cup \{\epsilon\}$, $i \in \{1 \dots r\}$. If $\tau_i \neq \rho_i$, then we say that τ_i *differs* from ρ_i . We distinguish among the following three types of differences.

Neither of these two symbols correspond to *no symbol* and the symbol of the first sequence corresponds to a different symbol of the second one, that is $\tau_i \neq \rho_i$ and $\tau_i \neq \epsilon$ and $\rho_i \neq \epsilon$. This type of difference is called *mismatch* (or *substitution*).

The symbol of the first sequence corresponds to *no symbol*

of the second sequence, that is $\tau_i \neq \epsilon$ and $\rho_i = \epsilon$. This type of difference is called *deletion*.

The symbol of the second sequence corresponds to *no symbol* of the first sequence, that is $\tau_i = \epsilon$ and $\rho_i \neq \epsilon$. This type of difference is called *insertion*.

Let $t = t_1 t_2 \dots t_n$ and $p = p_1 p_2 \dots p_m$ with $m < n$. We say that p occurs at position q of t with at most k -differences (or equivalently an *alignment* of p and t at position q with at most k -differences), if one can transform p into $t_q \dots t_{q+m-1}$ with at most k basic operations insertion, deletion and substitution.

The problem of string searching with k -differences is defined as follows: given a text $t = t_1 t_2 \dots t_n$, a pattern $p = p_1 p_2 \dots p_m$ and an integer k , find all occurrences of the pattern p in the text t with at most k -differences.

3 The Shift-Or Algorithm

The Shift-Or algorithm is used to solve exact string matching problems for a singular pattern and a singular text, developed by Baeza-Yates & Gonnet (1992). It is very fast in practice as well as easy to implement. Let R^0 be a bit array of size m . Vector R_i^0 is the value of the entire array R^0 after text character $y[i]$ has been processed. It contains information about all matches of prefixes of x that end at position i in the text ($0 \leq j \leq m - 1$):

$$R_i^0[j] = \begin{cases} 0 & \text{if } x[0 \dots j] = y[i - j, \dots, i] \\ 1 & \text{otherwise} \end{cases}$$

Therefore, $R_i^0[m - 1] = 0$ is equivalent to say that an exact occurrence of the pattern x ends at position i in y . The vector R_i^0 can be computed after R_{i-1}^0 by the following recurrence relation:

$$R_i^0[j] = \begin{cases} 0 & \text{if } R_{i-1}^0[j - 1] = 0 \text{ and } x[j] = y[i] \\ 1 & \text{otherwise} \end{cases} \quad \text{and} \\ R_i^0[0] = \begin{cases} 0 & \text{if } x[0] = y[i] \\ 1 & \text{otherwise} \end{cases}$$

The transition from R_{i-1}^0 to R_i^0 can be computed very fast as follows. For each $a \in \Sigma$, let Sa be a bit array of size m defined by: for $0 \leq j \leq m - 1$, $Sa[j] = 0$ iff $x[j] = a$. The array Sa denotes the positions of the character a in the pattern x . Each Sa can be preprocessed before the search. And the computation of R_i^0 reduces to two operations, SHIFT and OR: $R_i^0 = \text{SHIFT}(R_{i-1}^0) \text{ OR } Sy[i]$.

Assuming that the pattern length is not longer than the memory-word size of the machine, the time complexities of the preprocessing phase and searching phase are $O(|\Sigma| + m)$ and $O(n)$ respectively.

4 Modified Input Data

Searching for a pattern in a score is more complicated than searching for a pattern in a string because of the presence of the rhythm. Since the rhythm is as significant as the pitch for a melody, we will consider not only the pitch of the notes but also the rhythm. In order to adapt the Shift-Or algorithm to distributed matching problem within polyphonic sources, input data needs to be modified - by dividing all notes into the length of the shortest note present in the given data. This modification will be done very easily and takes a linear time.



Figure 1: Score and a pattern from Bach's Italian Concerto

Table 1 shows simple input data of the first bar of the example (Fig. 1). The data format is *(Pitch, Length)*. In this example, the shortest note is a quaver = length 1. Table 2 shows the modified input data of the bar. All notes are divided into quavers.

voice1	(72,2),(70,2),(69,4)
voice2	(0,1),(67,2),(67,1),(0,1),(67,1),(65,1),(64,1)
voice3	(52,2),(48,2),(53,2),(48,2)
pattern	(72,1),(67,1),(70,1),(67,1),(69,1),(67,1),(65,1),(64,1)

Table 1: Simple input data of a bar from Bach's Italian Concerto

i	0	1	2	3	4	5	6	7
v1	72	72	70	70	69	69	69	69
v2	0	67	67	67	0	67	65	64
v3	52	52	48	48	53	53	48	48
pa.	72	67	70	67	69	67	65	64

Table 2: Modified input data of a bar from Bach's Italian Concerto

In this example, the total length of the original score, in other words, the number of notes in the score is: $N = 3$ (Voice1) + 7 (Voice2) + 4 (Voice3) = 14. And the length of the original pattern is: $M = 8$. The total length of the modified score is: $N' = n \times v = 8 \times 3 = 24$, where n is the number of the notes in each modified voice, and v is the number of the voices. And the length of the mod-

$\alpha N (\alpha \geq 1)$, and $m = O(M) = \beta M (\beta \geq 1)$. In the example (Bach's Italian Concerto), $\alpha = 1.7$ and $\beta = 1.0$. However, the general values of α and β cannot be shown, because the relationship between the length of the original input data and the length of the modified input data depends entirely on the lengths of the given notes and the shortest note.

We can now apply the Shift-Or algorithm to this modified melodic data because the data can be handled like string data - all notes are the same length. However, the Shift-Or algorithm has to be modified for this problem, and the length of a modified pattern normally cannot be more than 64 (long long int). Since the original data is divided into the length of the shortest note, the algorithm does not know which modified note belongs to which original note. This allows the algorithm to find slightly different patterns from the original pattern as long as the sequence of the pitch and each length is the same, and this will affect the value of k (Levenshtein distance) of approximate distributed matching. For instance, the following occurrences would be found as exact matches (Fig. 2):



Figure 2: Possible exact matches (Bach's Italian Concerto)

5 Exact Distributed Matching Problem

	i	0	1	2	3	4	5	6	7
v	1	72	72	70	70	69	69	69	69
	2	0	67	67	67	0	67	65	64
	3	52	52	48	48	53	53	48	48
p	72	0	0	1	1	1	1	1	1
	67	1	0	0	1	1	1	1	1
	70	1	1	0	0	1	1	1	1
	67	1	1	1	0	1	1	1	1
	69	1	1	1	1	0	1	1	1
	67	1	1	1	1	1	0	1	1
	65	1	1	1	1	1	1	0	1
	64	1	1	1	1	1	1	1	0

Table 3: R_i^0 (Bach's Italian Concerto)

The Shift-Or algorithm can be adapted without any modifications to searching monophonic music patterns within

to solve exact distributed matching problems in polyphonic sources. If there are v voices:

$$R_i^0 = \text{SHIFT}(R_{i-1}^0) \text{ OR } (Sy_{[0][i]} \text{ AND } Sy_{[1][i]} \text{ AND } \dots \text{ AND } Sy_{[v-1][i]}).$$

Table 3 shows $R_i^0 = \text{SHIFT}(R_{i-1}^0) \text{ OR } (Sy_{[0][i]} \text{ AND } Sy_{[1][i]} \text{ AND } Sy_{[2][i]})$ in the previous example (Bach's Italian Concerto).

The modified algorithm is shown in Fig. 3, and it runs very fast (Fig. 4). Time complexities of preprocessing phase and searching phase are $O(|\Sigma| + m)$ and $O(N)$ respectively, where m is the length of the modified pattern, $|\Sigma|$ is the size of the alphabet, N is the length of the original score. The alphabet size will be 88 (the number of keys of the piano) plus 1 (rest). We need to preprocess 89 data with the modified pattern data.

```

begin
  for each  $a \in \Sigma$  do  $T[a] \leftarrow 2^m - 1$ 
  for  $j \leftarrow 1$  to  $m$  do  $T[p[j]] \leftarrow T[p[j]] - 2^{j-1}$ ;
   $E \leftarrow 2^m - 1$ ; //  $E$  denotes a bit-vector
  for  $i \leftarrow 1$  to  $n$  do
     $E \leftarrow \text{shiftright}(E) | (T[t_{[1][i]}] \& T[t_{[2][i]}] \& \dots \& T[t_{[v][i]}])$ ;
    if  $E.m = 0$  then write( $i$ );
end

```

Figure 3: Modified Shift-Or Algorithm for Exact Distributed Matching Problem

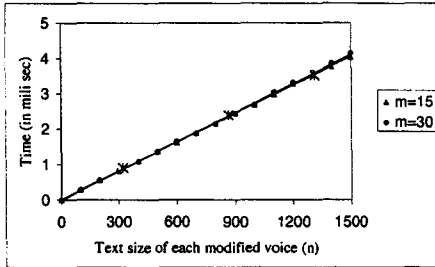


Figure 4: Timing curves for the modified Shift-Or Algorithm (4 voices) *n=320: Die Kunst der Fuge (Bach) excerpt, *n=864: Flute quartet in D (Mozart) excerpt, *n=1311: String Quartet No.16 (Beethoven) excerpt. Using a SUN Ultra Enterprise 300MHz running Solaris Unix.

6 Approximate Distributed Matching Problem

6.1 Wu-Manber Algorithm

There is a modified Shift-Or algorithm, called the Wu-Manber algorithm, which can solve approximate string

insertion, deletion, and substitution. The algorithm maintains $k + 1$ bit arrays R^0, R^1, \dots, R^k . The vector R^0 is maintained similarly as in the exact matching case. The other vectors are computed with the formula ($1 \leq j \leq k$): $R_i^j = (\text{SHIFT}(R_{i-1}^j) \text{ OR } Sy_{[i]}) \text{ AND } \text{SHIFT}(R_{i-1}^{j-1} \text{ AND } R_{i-1}^{j-1})$.

The time complexities of its preprocessing phase and searching phase are $O(|\Sigma| + m + k)$ and $O(kn)$ respectively.

6.2 Modified Wu-Manber Algorithm for Distributed Matching Problem

We modified the Wu-Manber algorithm in order to solve approximate distributed matching problems in polyphonic sources. If there are v voices:

$R_i^j = (\text{SHIFT}(R_{i-1}^j) \text{ OR } (Sy_{[0][i]} \text{ AND } Sy_{[1][i]} \text{ AND } \dots \text{ AND } Sy_{[v-1][i]})) \text{ AND } \text{SHIFT}(R_{i-1}^{j-1} \text{ AND } R_{i-1}^{j-1}) \text{ AND } R_{i-1}^{j-1}$. Fig. 5 shows the Modified Wu-Manber Algorithm, and it runs very fast (Fig. 6).

Time complexities of preprocessing and searching are $O(|\Sigma| + m + k)$ and $O((k + v)n) = O(N)$ respectively, where $|\Sigma|$ is the size of the alphabet, m is the length of the modified pattern, k is the Levenshtein distance allowed, v is the number of voices, n is the length of each modified voice, and N is the length of the original score. The value of k does not correspond to the Levenshtein distance between an original score and an original pattern, but the Levenshtein distance between a modified score and a modified pattern.

begin

```

  for each  $a \in \Sigma$  do  $T[a] \leftarrow 2^m - 1$ 
  for  $j \leftarrow 1$  to  $m$  do  $T[p[j]] \leftarrow T[p[j]] - 2^{j-1}$ ;
   $R_0^0 \leftarrow 2^m - 1$ ; //  $R$  denotes a bit-vector
  for  $h \leftarrow 1$  to  $k$  do  $R_0^h \leftarrow \text{shiftright}(R_0^{h-1})$ 
  for  $i \leftarrow 1$  to  $n$  do
    mask  $\leftarrow (T[t_{[1][i]}] \& T[t_{[2][i]}] \& \dots \& T[t_{[v][i]}])$ 
    for  $h \leftarrow 1$  to  $k$  do
       $R_i^h \leftarrow (\text{shiftright}(R_{i-1}^h) | \text{mask})$ 
       $\& \text{shiftright}(R_{i-1}^{h-1} \& R_{i-1}^{h-1}) \& R_{i-1}^{h-1}$ ;
    if  $R_i^k.m = 0$  then write( $i$ );
end

```

end

Figure 5: Modified Wu-Manber Algorithm for Approximate Distributed Matching Problem

6.3 Octave-displaced Matches

It often happens, for instance, that the cello plays a melody and the violin takes over or repeats it. In this case, the repeated melody is normally one or more octaves higher. Fig. 7 is an example from 4th movement of Beethoven's String Quartet No.16. To detect both occurrences, input data has to be modified and suppressed - divide the data by 12 and add 1 to the remainders (except 0 - rest). Then

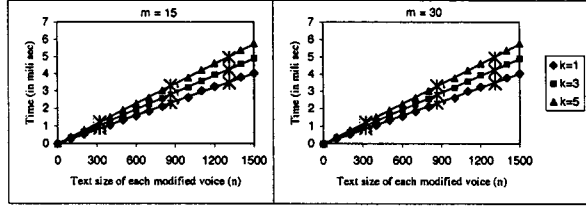


Figure 6: Timing curves for the modified Wu-Manber Algorithm (4 voices) *n=320: Die Kunst der Fuge (Bach) excerpt, *n=864: Flute quartet in D (Mozart) excerpt, *n=1311: String Quartet No.16 (Beethoven) excerpt. Using a SUN Ultra Enterprise 300MHz running Solaris Unix.

$C=1$, $C\# = 2, \dots, B=12$, and rest will stay 0. This modification will be done in linear time. In the previous example, the suppressed data of the violin and cello will be the same. Therefore, both patterns will be detected as exact matches. Needless to say, we need to modify the alphabet size and the data in the preprocessing phase as well. The alphabet size will be 13 (0 for rest, and 1,2,...,12 for C,C#,...,B) instead of 89.



Figure 7: Melodies from Beethoven's String Quartet No.16

7 Implementation

In this section, we implement the codes with actual music pieces. We have chosen the beginning of Beethoven's Symphony No.5 as a good example of distributed matching problem, and 34 bars from Beethoven's String Quartet No.16.

7.1 Beethoven's Symphony No.5 - 1st Movement

7.1.1 Score

Since clarinets and bassoons play the same notes as strings in the score (Fig. 8), and no other instruments play here, only the string section will be considered in order to simplify the process. In this example, the total length of the original score, in other words, the number of notes in the score is: $N=28$ (1st Violin) + 24 (2nd Violin) + 26 (Viola) + 15 (Cello) + 11 (Bass) = 104. As the shortest note is a quaver, the total length of the modified score is:



Figure 8: Score of the beginning of Beethoven's Symphony No.5

	Position
Exact Matching	35
Approx. Matching ($k = 1$)	34,35,36
Approx. Matching ($k = 2$)	33,34,35,36,37

Table 4: The output for the pattern and score (Beethoven's Symphony No.5)

notes in each modified voice, and v is the number of the voices. Therefore, $N' = O(N) = \alpha N = 2.5N$. Since one bar includes 4 quavers and the shortest note is a quaver, one bar includes 4 positions (for instance, position 60 corresponds to the first quaver of bar 16).

7.1.2 Pattern



Figure 9: A pattern for Beethoven's Symphony No.5

The length of the original pattern (Fig. 9) is: $M = 13$. Modified pattern: [0, 67, 67, 67, 63, 68, 68, 68, 67, 75, 75, 75, 72, 72, 72, 72]

The length of the modified pattern is: $m = 16$, as the shortest note is a quaver: $m = O(M) = \beta M = 1.2M$

7.1.3 Implementation

There is an exact match from position 20 to 35. The output (the last position of the occurrence) is shown in Table 4.

	Position
Exact	47(1stVn)
Approx.($k = 1$)	46,47(1stVn),48
Approx.($k = 2$)	45,46,47(1stVn),48,49

Table 5: The output for the pattern and score (Beethoven's String Quartet No.16)

7.2 Beethoven's String Quartet No.16 - 4th movement

7.2.1 Score

Fig. 10 shows a score from Beethoven's String Quartet No.16. In this example, the total length of the original score is: $N = 350$. As the shortest note is a crochets, the total length of the modified score is: $N' = n \times v = 544$. Therefore, $N' = O(N) = \alpha N = 1.6N$.



Figure 10: Score of the beginning of Beethoven's String Quartet No.16

7.2.2 Pattern

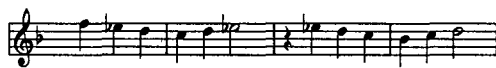


Figure 11: A pattern for Beethoven's String Quartet No.16

The length of the original pattern (Fig. 11) is: $M = 13$. Modified pattern: [77, 75, 74, 72, 74, 75, 75, 0, 75, 74, 72, 70, 72, 74, 74]

The length of the modified pattern is: $m = 15$, as the shortest note is a crochets: $m = O(M) = \beta M = 1.15M$.

7.2.3 Implementation

There is an exact match in the 1st Violin from position 33 to 47. The output (the last positions of the occurrences) is just as expected (Table 5).

7.2.4 Octave-displaced matching

There is a similar pattern (one octave lower and with one substitution) in the Cello from position 35 to 49 (Fig. 12). To detect this pattern too, we need to apply approximate octave-displaced matching program.

Suppressed pattern: [6, 4, 3, 1, 3, 4, 4, 0, 4, 3, 1, 11, 1, 3, 3]

The output is just as expected (Table 6).



Figure 12: A similar pattern

	Position
Exact	47(1stVn)
Approx.($k = 1$)	46,47(1stVn),48,49(Cello)
Approx.($k = 2$)	45,46,47(1stVn),48,49(Cello),50

Table 6: The output for the suppressed pattern and score (Beethoven's String Quartet No.16)

7.2.5 Transposed match

Now, all expected exact matches, approximate matches, (approximate) octave-displaced matches are found with no false matches. However, there is another important match in the Cello - transposed match. Fig. 13 shows the same motif (Cello, position 1-15) as the pattern, but not in the same key and with one substitution. To find transposed matches, create 11 transposed patterns by adding 1 to 11 to the suppressed pattern data except 0, and if the data exceeds 12, then subtract 12 from it. Run the program with all the 12 patterns.

The output is just as expected (Table 7). Now, we found all relevant occurrences with no false matches.



Figure 13: A transposed pattern

8 Asymptotic Time Complexity

Assuming $|\Sigma|$ and k are constant, and the word-size is larger than m , all distributed matching problems (exact, approximate, octave-displaced, and transposed) are solved

	Position
Exact	47(1stVn)
Approx.($k = 1$)	15(Cello),46,47(1stVn),48,49(Cello)
Approx.($k = 2$)	14,15(Cello),16,45,46, 47(1stVn),48,49(Cello),50

Table 7: The output for the transposed patterns (Beethoven's String Quartet No.16)

in $O(N + M)$ time.

N	Total length of an original score
M	The length of an original pattern
n	The number of notes in each modified voice
v	The number of voices
N'	The total length of a modified score
m	The length of a modified pattern
$ \Sigma $	13 for suppressed data, otherwise 89
k	Allowed Levenshtein distance

• **Exact Distributed Matching:**

$$O(N + M + |\Sigma|)$$

1. Modify a score and a pattern:
 $O(N' + m) = O(N + M)$
2. Shift-Or - Preprocessing:
 $O(|\Sigma| + m) = O(|\Sigma| + M)$
3. Shift-Or - Searching:
 $O(vn) = O(N)$

• **Approximate Distributed Matching:**

$$O(N + M + k + |\Sigma|)$$

1. Modify a score and a pattern:
 $O(N' + m) = O(N + M)$
2. Shift-Or - Preprocessing:
 $O(|\Sigma| + m + k) = O(|\Sigma| + M + k)$
3. Shift-Or - Searching:
 $O((v + k)n) = O(N + (kN)/v) = O(N)$

• **Exact Octave-displaced Distributed Matching:**

$$O(N + M + |\Sigma|)$$

1. Modify and suppress a score and a pattern:
 $O(N' + m) = O(N + M)$
2. Shift-Or - Preprocessing:
 $O(|\Sigma| + m) = O(|\Sigma| + M)$
3. Shift-Or - Searching:
 $O(vn) = O(N)$

• **Approximate Octave-displaced Distributed Matching:**

$$O(N + M + k + |\Sigma|)$$

1. Modify and suppress a score and a pattern:

2. Shift-Or - Preprocessing:

$$O(|\Sigma| + m + k) = O(|\Sigma| + M + k)$$

3. Shift-Or - Searching:

$$O((v + k)n) = O(N + (kN)/v) = O(N)$$

• **Exact Transposed Distributed Matching:**

$$O(N + M + |\Sigma|)$$

1. Modify and suppress a score and a pattern:
 $O(N' + m) = O(N + M)$
2. Produce 11 transposed patterns:
 $O(11m) = O(M)$
3. Shift-Or - Preprocessing:
 $12 \times O(|\Sigma| + m) = O(|\Sigma| + M)$
4. Shift-Or - Searching:
 $12 \times O(vn) = O(N)$

• **Approximate Transposed Distributed Matching:**

$$O(N + M + k + |\Sigma|)$$

1. Modify and suppress a score and a pattern:
 $O(N' + m) = O(N + M)$
2. Produce 11 transposed patterns:
 $O(11m) = O(M)$
3. Shift-Or - Preprocessing:
 $12 \times O(|\Sigma| + m + k) = O(|\Sigma| + M + k)$
4. Shift-Or - Searching:
 $12 \times O((v + k)n) = O(12N + (12kN)/v) = O(N)$

9 Conclusion and Limitations

Using the Shift-Or algorithm, distributed matching problems (exact, approximate, octave-displaced, and transposed matching) for musical melodic recognition are solved in linear time. The functions (the modified Shift-Or algorithm and modified Wu-Manber algorithm) run very fast and work well with actual musical pieces with few false matches. However, there are some limitations. Since the pitch and the rhythm are considered, we had to make the length of modified patterns longer, even though it cannot exceed 64. 64 should be enough normally, but if triplets and semi-quavers coexist in a score, for instance, a crochet could be divided into 12 notes. To reduce the length, we could consider *musical events* to be a list of notes which begin to play at the same time. However, the rhythm cannot be considered, and the Shift-Or Algorithm may not work well on it.

These codes cannot identify one major form of patterns in music, namely harmonic progressions. This limitation is apparent, for example, when considering variation form (which by definition is pattern-based). An example of this might be the opening movement of Mozart's A-major Piano Sonata, which is a set of variations on a simple melody. Whilst musicians can recognize a harmonic rela-

Shift-Or algorithm is unable to identify this relationship no matter how large is the value of k .

Acknowledgements

Costas S. Iliopoulos was partially supported by a Marie Curie Fellowship and Royal Society, Wellcome Foundation and NATO grants.

References

- A. Apostolico, Z. Galil. Pattern Matching Algorithms. Oxford University Press, 1997.
- R. A. Baeza-Yates and G. H. Gonnet. A new approach to Text searching. CACM, 74-82, 1992.
- R. A. Baeza-Yates and G. Navarro. A faster algorithm for approximate string matching. Proceedings of the 7th Symposium on Combinatorial Pattern Matching, LNCS, 1075:1-23, Springer-Verlag, New York, 1996.
- E. Cambouropoulos. A general pitch interval representation, Theory and applications. Journal of New Music Research, 25:231-251, 1996.
- C. Charras and T. Lecroq. Exact String Matching Algorithms. [<http://www-igm.univ-mlv.fr/lecroq/string/>], 1997.
- R. Cole, R. Hariharan, and P. Indyk. Tree pattern matching and subset matching in deterministic $O(n \log^3 m)$ time. Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 245-254, 1999.
- R. Cole, C. S. Iliopoulos, T. Lecroq, W. Plandowski, and W. Rytter. Relations between delta-matching and matching with don't care symbols: delta-distinguishing morphisms. Proceedings 12-th Australasian Workshop on Combinatorial Algorithms, 2001
- T. Crawford, C. S. Iliopoulos, and R. Raman. String Matching Techniques for Musical Similarity and Melodic Recognition. Computing in Musicology, 11:73-100, 1998.
- M. Crochemore and T. Lecroq. Pattern Matching and Text Compression Algorithms, in A.B. Tucker Jr, ed. The Computer Science and Engineering Handbook, CRC Press, Boca Raton, 1997.
- M. Dovey and T. Crawford. Heuristic Models of Relevance in Searching Polyphonic Music. Diderot Forum on Mathematics and Music, Vienna, December 2-4, 111-123, 1999.
- M. Dovey. A technique for "regular expression" style searching in polyphonic music. 2001.
- Z. Galil and K. Park. An improved algorithm for approximate string matching. SIAM Journal on Computing, 19:989-999, 1990.
- J. Holub, C. S. Iliopoulos, B. Melichar, and L. Mouchard. Distributed Pattern Matching Using Finite Automata. Journal of Automata, Languages and Combinatorics 6, 2:191-204, 2001.
- G. M. Landau and U. Vishkin. Fast string matching with k difference. Journal of Computer and Systems Sciences, 37:63-78, 1998.
- K. Lemström. String Matching Techniques for Music Retrieval. University of Helsinki, 2000.
- W. J. Masek and M. S. Paterson. A Fast algorithm for computing string edit distances. J. Comput. Sy. Sci., 20:18-31, 1980.
- D. Meredith, G. A. Wiggins, and K. Lemström. Pattern induction and matching in polyphonic music and other multi-dimensional datasets. Proceeding of the 5th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI2001), July 22-25, Orlando, FL., X:61-66, 2001.
- J. Pinzon. Algorithms for Computing Approximate Repetitions in Musical Sequences. Thesis, 1999.
- P. H. Seller. The theory and computation of evolutionary distances: Pattern recognition. Journal of Algorithms, 1:359-373, 1980.
- E. Ukkonen. Finding approximate patterns in strings. Journal of Algorithms, 6:132-137, 1985.
- S. Wu and U. Manber. Fast text searching allowing errors. CACM, 35:83-91, 1992.
- S. Wu, U. Manber, and G. Myers. A subquadratic algorithm for approximate limited expression matching. Algorithmica, 15:50-67, 1996.
- A. H. Wright. Approximate string matching using within-word parallelism. Soft. Pract. Exper., 24:337-362, 1994.

A psychosocial model for the evolution of aesthetic patterns

Thibaud de Souza¹ and Tatiana Kalganova
Brunel University, Uxbridge UB8 3PH

¹ee00ttd@brunel.ac.uk; ²tatiana.kalganova@brunel.ac.uk

Abstract

This paper describes an original attempt to evolve aesthetic patterns by integrating the rules of colour psychology into a multi-agent evolutionary model. The system uses the principles of evolution to determine social relationships between agents. Communication plays an important role in the evolution of social behaviour. In our case the exchange of information between agents determines their behavioural characteristics. The interaction between agents and their social behaviour may be controlled and monitored using real-time image animation technique.

1 Introduction

It is well known that it is difficult to guide the evolution of meaningful aesthetic patterns. Important problems associated to this include the design of systems with generic representational capabilities, the question of aesthetic judgement and knowledge integration (Machado et al 1996).

The use of artificial intelligence in evolution of artwork has been researched actively during the last decade. Thus, experiments described earlier involve manual guidance of evolution (Rooke, 2001). Mechanisms for rearranging ready-made forms have been presented in (Soddu, 2001). Behavioural agents have been used to evolve images of virtual creatures (biomorphs), through the use of genetic algorithm (Dawkins, 1987). However, in this case again, manual selection has been involved.

Recently the modelling of emotions has attracted much attention from researchers. Using this technique, believable agents have been created (El-Nasr et al 1998;). At the same time it has started to be considered an essential component of intelligence (Gershenson, 1999; El-Nasr et al 1998).

Following some insights provided in (Taylor, 2001), we have attempted to evolve aesthetic patterns without manual intervention. We suggest that automatic evolution of aesthetic forms may be achieved by combining the following:

1- A generative system with sound potential in terms of the variety and level of organisation of created forms.

2- Behavioural scripts able to manipulate the data structures contained in the system

3- Encoded rules binding the behavioural scripts to the perceived psychological qualities of the produced forms.

A system binding behavioural agents to the perceived psychological values associated with colour was designed and implemented in order to test this hypothesis. In this paper, the design of the system is exposed, along with preliminary experimental results.

2. Modelling social behaviour

The purpose of the system is to evolve scripts describing the behaviour of individuals within social groups. Each individual is described using parameters that define its physical and emotional states and a behavioural script. Physical states of agents determine their emotional state. Emotional states control the execution of an individual's script. Behavioural script defines the interactions between agents.

Relations between individuals have been modelled at 3 distinct levels:

- The emotional states of an individual are derived from the colour of its neighbours using the rules of colour psychology.
- Informational and material transfers are performed between individuals in the form of data, code, colour, density and physical links exchange.
- Social units are modelled using directed links binding individuals. Such links constrain the relative positions of individuals in space while facilitating informational exchanges and amplifying individual contributions to local emotional climates.

The location of agents is constrained within spherical bounds. The graphical patterns created reflect the evolution of individual and group attributes over time.

The system allows controlling interactively the rules that constrain the behaviour of agents and the evolution of their physical and emotional states.

3. Agent characteristics

In real life, individuals may be described by their physical and psychological states.

3.1 Physical states

In our system, physical attributes include location, colour, speed and density (see Table 1). The speed vector is decomposed into 3 distinct component vectors representing the agent's speed, the pull exerted by physical links (spring vector) and a variable amount of random noise.

Table 1: Agent physical attributes

location	speed												colour	density	link array
	x0	y0	z0	x1	y1	z1	x2	y2	z2	x3	y3	z3			
x0	0.0	0.0	0.0	x1	0.0	0.0	x2	0.0	0.0	x3	0.0	0.0	red	0.000	0.000

The location of each individual is initialised randomly within specified bounds. At each execution cycle the location is updated using the follow rule:

$$location = location + s$$

where s is calculated as a sum of cell speed, spring vector and noise vector.

The cell's speed is updated using the cell's target location, friction and acceleration parameters:

$$speed = (speed + A_{acceleration} \cdot u) \cdot (1 - A_{friction})$$

where u is a normalised vector representing the direction of target location; $A_{acceleration}$ is an agent acceleration that is constant; $A_{friction}$ is an agent friction that is constant.

The spring vector is the sum of the spring forces Fs_i generated by each physical links:

$$S = (S + \sum_{0 \leq i < N_{links}} Fs_i) \cdot (1 - S_{friction})$$

where $S_{friction}$ is a spring friction that is constant; N_{links} is the number of physical links, Fs_i is a spring

forces that is calculated according to the following equation:

$$Fs_i = v \cdot (d_i - D_i) \cdot S_{power}$$

where v is a normalised vector representing the direction of the physical link; d_i is current length of physical link; D_i is reference length of the physical link; S_{power} is the spring power constant.

The noise vector can be jittered randomly with a specified probability.

3.2 Colour coding scheme for modelling emotions

The psychological state of individuals has been modelled by describing the influence of the psychological climate produced by colours on individuals. Fig. 1 illustrates the relationships between agents and their emotions. The expressions of agents are derived from their colours. The expression values contribute to the emotion values of neighbouring individuals (see Fig. 2).

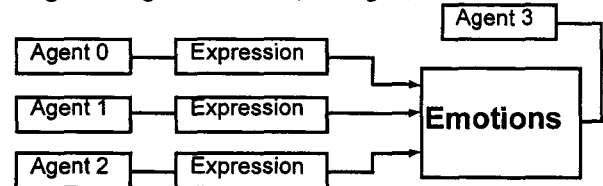


Figure 1: Deriving agents' emotions from neighbouring agents' expressions.

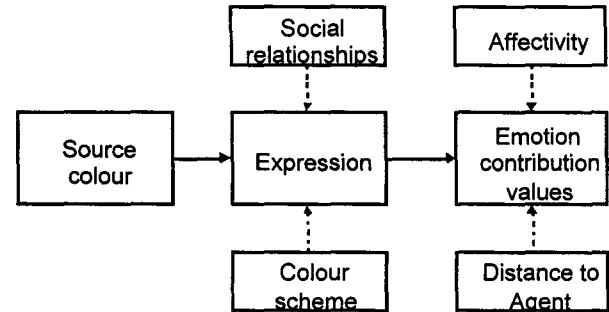


Figure 2: Evaluation of emotions

For each individual expressions and emotions are modelled. For any individual, values for 8 basic emotions and expressions are stored in arrays of decimal numbers in the range [0, 1] (see Table 2). The higher coefficient, the stronger the feeling.

Table 2: Psychological attributes of agent, where $E_i, M_j \in [0,1]$.

	love	seduction	despair	emptiness	sadness	calm	happiness	anger
expression-ref	E0	E1	E2	E3	E4	E5	E6	E7
reference-ref	M0	M1	M2	M3	M4	M5	M6	M7

The expression of an individual is derived from its current colour using the rules of colour psychology. The intensity value for each expression component X is calculated as a function of the agent's colour X and the expression component reference colour X .



All expression components are then rescaled so that the maximum expression component value is 1.0 and the minimum is 0.0.

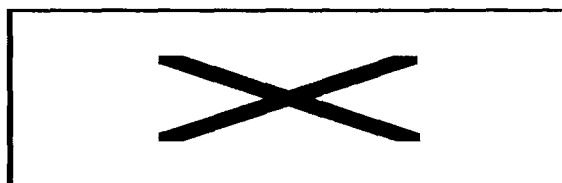
The overall expression intensity value is calculated according to the reference sizes of the physical links X established by the individual with its neighbours (see Table 3).

Table 3: Calculation of expressions overall intensity

n. of links	expression intensity value
0	E_0
1	E_0^2
N	$\pi * (\sum D_i / N)^2 * E_0$

The chosen mapping between emotions and colour is exposed in Fig. 3.

The emotions of individuals X are derived from the expressions of neighbours according to the equation:



where X are emotion calculus coefficients that are constants; X is the number of neighbours; X is the agent affectivity.

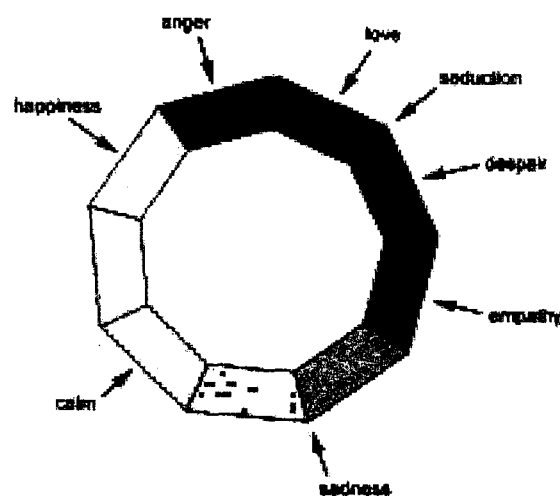


Figure 3: Colour coding scheme

3.3 Behavioural scripts

Each agent holds a thread that will execute the agent's behavioural script. Execution threads use reference and scheduling to perform operation. Every time invoked, an agent's thread performs zero, one or more operations before releasing control.

Behavioural scripts consist in operation sequences. Each operation is identified by a descriptor. The descriptor specifies the following parameters:

- an operation identifier (opcode);
- a ratio used by specific operations;
- 4 Boolean values used by Boolean logic operators;
- program and memory pointers used to address specific areas in cell memory;
- cell addressing parameters;
- one identifier controlling write access privileges.

Scripts may manipulate data stored in an individual's memory.

Behavioural scripts are initialised semi-randomly. Operators that play a crucial role in the evolution of the system are seeded in higher proportion. Memory buffers are initialised randomly.

At each execution cycle, an execution thread invokes one or more instructions. The execution thread releases control when one of the following occurs:

- The latest operator invoked specified a non-zero execution delay or had its *break* parameter set
- The program pointer is out of range.
- A stop operator was executed.

Execution may be controlled using a range of parameters that will influence the evolution and quality of the produced system (see Table 4).

Table 4: Execution control parameters

enable write access modifiers	Allows scripts to protect memory or code fragments in writing.
progressive data transfer checking;	Controls whether data transfer operations validity is checked once per transfer or once for each transferred memory buffer
progressive program memory transfers	Controls whether program code transfer operations validity is checked once per transfer or once for each transferred operator
allow data transfer offsets;	Controls whether an offset address should be used when transferring data from cell to cell
allow program transfer offsets	Controls whether an offset address should be used when transferring code from cell to cell
action threshold values A0,A1	Used to determine success of failure of action operators, based on the distance from the caller to the target cell.
perception threshold values P0, P1	Used to determine success of failure of perception operators, based on the distance from the caller to the target cell.

Each operator has a number of associated parameters: operator cost, break parameter, emotion code and emotion reference value (Table 5).

3.3 Operators

In the systems a number of operators has been defined according to the evolution of the system. Thus, one of the introduced operators is the transfer operator. This operator can transfer colour, link material or density to target agent using specified ratio. The operator has been designed in such way that no material can be added or removed to/from the system. If some material has been transferred from one agent to another, then the same amount of material will be subtracted from the source agent. Another operator that has been introduced in the system is a motion operator. The purpose of this operator is to identify the target location for the agent. The link operator allows establishing the physical links between agents. The specified value of link material can be allocated using an allocation operator. The allocation is carried out between two previously linked agents. Active agents are allowed to exchange information with their neighbourhood. In this case distance, colour and location of agent are taken into account. The enquiry is used in order to establish contacts with agents in neighbourhood and if the answer is positive the exchange of information takes place.

Table 5: Operation attributes

cost	Represents the amount of energy needed to execute operator.
break parameter	Determines whether an agent's thread should release control after execution of the associated operator.
execution delay	The time (in frames) necessary to complete execution.
emotion code	Successful execution of the given operator will be dependant on the value of one specified emotion
emotion reference value	The threshold value that the given emotion should have for the given operator to always execute successfully.

3.4 Evaluation criteria

The system potentially contains self-reproducing behavioural patterns involving one or more individuals. Evolutionary pressure arises primarily as a result of memory resources limitations. The fittest behavioural patterns use the given instruction set advantageously to reproduce faster and more accurately according to the physical and psychological rules governing the system.

4 Graphical representation

The evolved patterns are represented in real-time. Images can be produced by rendering the system incrementally over time. Images contain information about agents and their physical links.

Fig. 4 illustrates an example of image produced after several generations. In this image the black tracks represent agent trajectories. The agents that established links with neighbourhood are drawn as polygons. The colours of polygons interpolate the colours of individuals. The presented picture is at an early stage of evolution. The distribution of colour is still random. This shows that the system has not evolved any specific emotional climate. One can notice that there are only few links established between agents. This is identified by the number of polygons in the presented image.



Figure 4: A sample output image

Physical environment and program execution parameters may be modified interactively. The user may tune the visual output using a range of parameters and navigate inside the system. These parameters include whether the links should be drawn or not, whether the neighbourhood should be drawn or not, etc.. The Viewing camera may be linked to one of the agents' location.

A comparison between Fig. 4 and Fig. 5 demonstrates the use of different viewpoints.



Fig. 5: Close up

5. Experimental results

The system has been implemented using C++. A number of experiments have been set up in order to test the technical and aesthetic qualities of the system. A population of one thousand of agents has been used in the performed series of experiments. The analysis of experiments shows the evolutionary qualities of the system, as well as its ability to produce a wide range of aesthetically pleasing visual patterns. Two main series of experiments have been carried out. In the first one the parameters of the system have been investigated in detail. In the second one the aesthetic qualities of the produced images have been evaluated by an expert in the field.

5.1 Numerical analysis

In order to identify how the system behaves depending on the used type of constraints, a number of experiments have been carried out. The purpose of the experiment is to illustrate how the system will behave with and without colour constraints. First of all we have been interested in the evolution of operator distribution in the system (Experiment A). In this case no colour scheme has been applied. In the second experiment, colour constraints have been applied (Experiment B). In this case one psychological colour based constraint has been applied to the copy and get cell reference operators.

Table 6: The parameters of the system
environment parameters

environment parameters

number of agents	768
link avail/cell	0.03
randomise colour	no
seed radius	0.25
world radius	1.50
reduced operator set ratio	0.25
agent acceleration	0.008
agent friction	0.001
agent max. speed	0.05
target threshold	0.05
action thresholds A0,A1	0.3,0.5
perception thresholds P0,P1	0.3,0.9
stamina recovery	0.01
environment depth	0.4
spring power	0.08
spring friction	0.6
spring max speed	0.045
noise amount	0.001
noise frequency	0.001
noise max speed	0.01
execution control	
use write protection	yes
progressive data transfer checking	yes
progressive program transfer checking	
data transfer offset	yes
program transfer offset	yes

The operators were set to warrant execution only for love emotion values superior to 0.6.

The initial values of parameter used in this experiment are shown in Table 6. The experimental results obtained are plotted in the graphs shown in Fig. 6 (Experiment A) and Fig. 7 (Experiment B). The analysis of obtained results shows the following :

- In both cases, evolution arises and can be detected by the characteristic shape of the graphs.
- The distribution of operators evolves more slowly when additional colour constraints are applied.
- When colour constraints are applied, there is an increase in the proportion of arithmetic operators.
- It can be clearly seen that the copy operators is used more often if there are no colour constraints.
- The transfer link material is used less in Experiment B.
- The distribution of operators slows down with increase of number of generations.

Analysis of the results of these experiments shows that the use of colour-based constraints influences the overall performance of the system.

5.2 Aesthetic qualities of the produced images

Images produced after 10000 generations have been presented to an expert in the area of graphic design to identify the strengths and weaknesses of the generated images. The evaluation of the images has been focused on the evocative power and the dynamic qualities of the images.

8 images have been presented to the expert in graphic design. Some of these images are reproduced in Fig. 8 – Fig. 10. Fig. 8 and Fig. 9 present the evaluated images as well as the comments of the expert. It is interesting to note that associations with Van Gogh have been drawn with one of the presented images. At the same time another examined image has been associated with the works of Balla (not shown).

The overall impression is that “most images have a fractured quality to them and tonally occupy a similar amount of variation – all the images share a similar amount of activity except two depicted in Fig. 8 and Fig. 10 which seem to be more active. Apart of Fig. 8 and several others they look like abstract prints ... The one I find hardest to look at is Fig. 8”.

Based on the comments of expert one can conclude that the produced, graphical patterns are novel, original and inspiring.

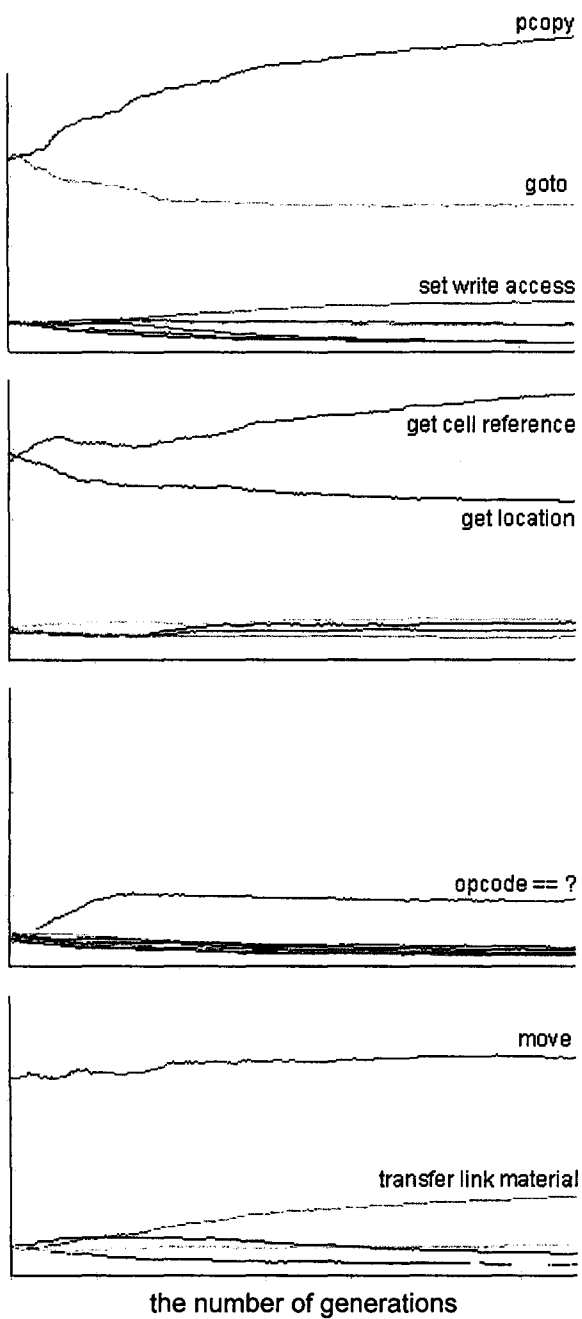


Figure 6: Experiment A: Distribution of operators over time without colour constraints. The horizontal axis represents the number of generations and the vertical axis represents the number of operators used in the system.

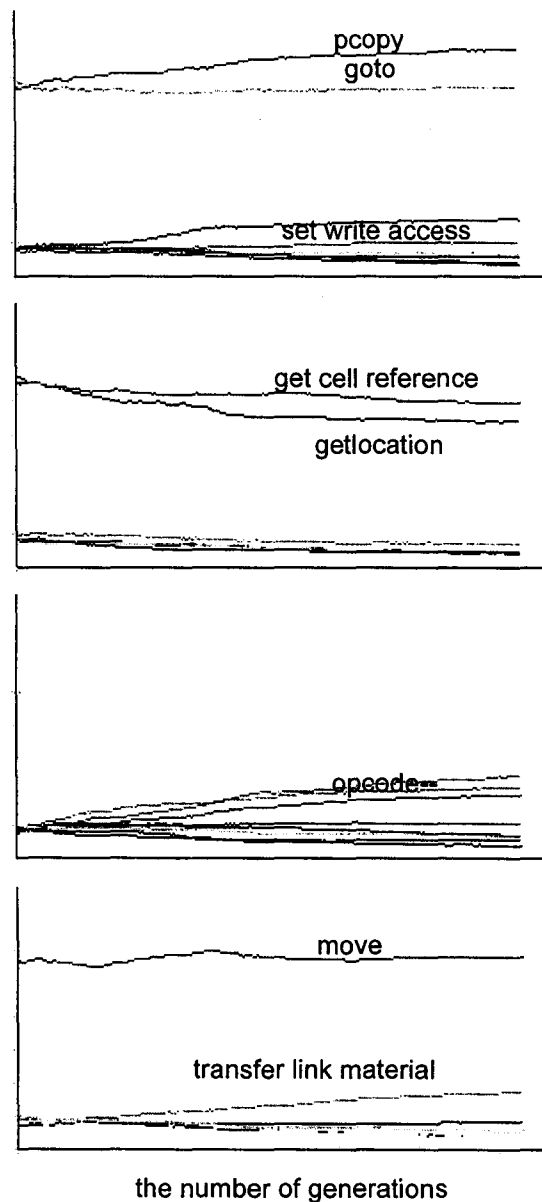


Figure 7: Experiment B: Distribution of operators over time with colour constraints. The horizontal axis represents the number of generations and the vertical axis represents the number of operators used in the system.



"A scratchy image, looking a little like a Van Gogh of a lave flow. Quite explosive but a cooler explosion - a steam geyser rather than a lave eruption. There is a sense of clashing and collision here between the linearity of the red (lower right) and the more chaotic greens and blues of the upper left."

Figure 8. The first selected image, "Domain"



"A forest, a multi-layered image that does not seem to have a "ground" or a base on which the images are constructed, there is the sense of a continuing, scalable environment. It all seems very organic and natural. A "flat" composition with a rather uniform emphasis (hence the feeling of no foreground and no background just a continuing space)."

Figure 9. The second selected image



Figure 10: The third selected "active" image, "Ice skating"



"Looking a mountain through frosted glass, its too tonally uniform to be mist. The more you examine it the more colourful it becomes and the more physical depth is perceived.... Turning it back to the correct orientation it reveals an eye in the centre, it is more sinister. Again it is top heavy, fractures, crystalline."

Figure 11. The fourth selected image



Figure 12. Generated image

A number of generated images produce very strong associations with real life. Thus, it has been noticed that the image depicted in Fig. 12 reminds a lot of the relaxing sea life. Dynamic motion inside the picture can be seen very clearly.

6. Future work

Extended analysis of the system may explore the spatial migration of passive and operational knowledge within the system and attempt to identify and understand specific behavioural patterns.

The rules of colour psychology should be combined with an anthropomorphic approach to space and motion to evolve appealing graphic configurations.

7. Summary

In this paper, a psychosocial model for the automated evolution of aesthetic patterns has been presented. A colour-coded scheme has been used to constrain the social behaviour of individuals. Experimental results demonstrate the influence of colour-based constraints on the evolution of behavioural and graphic patterns. An expert in graphic design has examined several produced images. The quality in terms of colour harmony and composition has been demonstrated. The dynamic is one of the specific features of generated images. The experimental results show the dominance of different colours in different neighbourhood. The produced images display aesthetic qualities in terms of colour harmony.

Acknowledgements

We would like to thank Leon Cruickshank from Brunel University for his evaluation of the produced images and anonymous reviews for their comments. The first author would like to thank Prof. John Stonham for advice and friends for kind support given during preparation of this paper.

References

- Dawkins, R. The blind watchmaker. W.W. Norton & company, Inc., 1987.
- El-Nasr, M.S. and M. Skubic. A Fuzzy Emotional Agent for Decision-Making in a Mobil Robot, *Proc. of Fuzz-IEEE 98*, 1998.
- El-Nasr, M.S. and J. Yen, Agents, Emotional Intelligence and Fuzzy Logic, *Proc. of the 17th Annual Meeting of the North American Fuzzy Information*, 1998.
- Gershenson, C. Modelling Emotions with Multidimensional Logic. *Proc. of the 18th Int. Conference of the North American Fuzzy Information Processing Society (NAFIPS '99)*, pp. 42-46. New York City, NY, 1999.
- Machado, P., Cardoso, A., Generation and Evaluation of Artworks. *Proc. of the 1st European Workshop on Cognitive Modeling, CM'96*, in Schmid, Krems and Wysotzli (Eds.), TR 96-39, Technische Universität Berlin, Germany, 1996.
- Rooke, S. Eons of Genetically Evolved Algorithmic Images. In *"Creative Evolutionary Systems"*. Eds: P.J. Bentley, D.W. Corne, published by Morgan Kauffman, 2001.
- Soddu, C. Recognizability of idea. In *"Creative Evolutionary Systems"*. Eds: P.J. Bentley, D.W. Corne, published by Morgan Kauffman, 2001.
- Taylor, T. Creativity in Evolution. In *"Creative Evolutionary Systems"*. Eds: P.J. Bentley, D.W. Corne, published by Morgan Kauffman, 2001.

Evolution of Musical Motifs in Polyphonic Passages

Costas S. Iliopoulos^{*†}; Kjell Lemström[°]; Mohammed Niyad^{*}; Yoan J. Pinzón[‡]

^{*} Department of Computer Science, King's College, London.

[†] School of Computing, Curtin University of Technology, WA.

[°] Department of Computer Science, University of Helsinki, Finland.

[‡] Facultad de Ingeniería de Sistemas, Universidad Autónoma de Bucaramanga, Colombia
{csi,niyadm}@dcs.kcl.ac.uk;klemstro@cs.helsinki.fi;ypinzon@bumanga.unab.edu.co

Abstract

In this paper we consider the problem of *motif evolution in polyphonic musical sequences*. A related problem, where a set of sequences of notes (one sequence for a voice) and a pattern is given, is to find whether approximate occurrences of the pattern occur distributed across the sequences (Holub et al., 1999; Lemström and Tarhio, 2000). Formally, this related problem is as follows: given a set t of h strings (each representing a *voice*) $t^i = t_1^i, \dots, t_n^i$, $i \in \{1..h\}$, for some constant h and a pattern $p = p_1, \dots, p_m$, we say that p occurs at position j of t if $p_1 = t_j^{i_1}, p_2 = t_{j+1}^{i_2}, \dots, p_m = t_{j+m-1}^{i_m}$ for some $\{i_1, \dots, i_m\} \in \{1..h\}$. Our problem of finding evolutionary chains is defined as follows: given a set t of h strings t^i (the *target*), for some constant h and a *motif* p , find whether there exists a sequence $u_1 = p, u_2, \dots, u_\ell$ occurring in the target t such that u_{j+1} occurs to the right of u_j in t and for any given $j \in \{1..\ell - 1\}$, u_j and u_{j+1} are similar enough, i.e., they do not differ more than by a certain number of basic operations — insertions, deletions and substitutions. In this paper, we consider several variants of the evolutionary chain problem and present efficient algorithms solving them.

1 Introduction

This paper is focused on a set of string pattern-matching problems which arise in music analysis (Mongeau and Sankoff, 1990; Stech, 1981), musical information retrieval (Lemström, 2000) and molecular sequence analysis (Gusfield, 1997). A musical score can be viewed as a string: at a very rudimentary level the alphabet could simply comprise the pitch names of the western music notation, or at a more complex level, we could use the GPIR representation of Cambouropoulos (1996a,b) as the basis of an alphabet. It is generally agreed in musicology and music psychology, that one of the most important phases in understanding a musical work is to identify the significant repetitions. The capability of identifying such repetitions would have a direct impact on automated music analysis and music information retrieval. The definition of musical repetition, however, is very vague. In its most restricted form, repetitions may be defined as excerpts being exactly similar (by considering intervals instead of absolute pitch values, repetitions that are merely transposed fall into this category of repetitions, as well).

Exact repetitions have been studied extensively. Such repetitions may either appear as distinct substrings (Apostolico, 1983; Iliopoulos et al., 1996b; Landau and Schmidt, 1993; Main and Lorentz, 1984; Myers and Kannan, 1993) or they may overlap (Berkman et al., 1996; Iliopoulos

and Mouchard, 1999; Iliopoulos et al., 1996a; Moore and Smyth, 1994). A natural extension of the repetition problem is to allow the presence of errors. Although this makes the task more challenging, it is usually much more pertinent approach, which is the case, especially, when dealing with music. In approximate pattern matching, error tolerance is achieved by introducing a similarity measure (such as *edit* or *Hamming distance*, see e.g. (Crochemore and Rytter, 1994)) and a threshold variable k indicating the allowed tolerance to errors. A repeated substring may be subject to other constraints (e.g., it may be required to be of at least a certain length) and some invariances, as well.

Efficient algorithms for computing the approximate repetitions are not only relevant to our music task at hand. Instead, they are directly applicable also, for instance, to molecular biology (Fischetti et al., 1992; Karlin et al., 1988; Milosavljevic and Jurka, 1993) and in particular in DNA sequencing by hybridization (Pevzner and Feldman, 1993), reconstruction of DNA sequences from known DNA fragments (Schmidt, 1994; Skiena and Sundaram, 1995), in human organ and bone marrow transplantation as well as the determination of evolutionary trees among distinct species (Schmidt, 1994).

In this paper, we study a certain modification of the approximate repetition problem, namely the evolution of

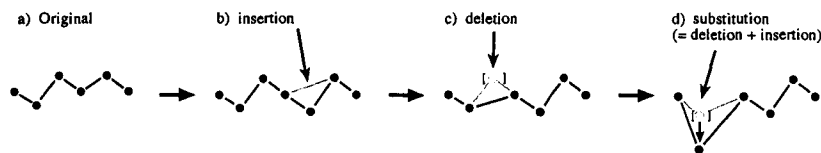


Figure 1: An example of an evolving motif and of local editions required to trace the gradual changes.

a monophonic¹ musical motif in a target of polyphonic music². A simplistic case of the problem, when the target is also monophonic, is defined as follows: Given a string t representing the target (in combinatorial pattern matching often called the *text*) and a motif p , find whether there exists a sequence $u_1 = p, u_2, \dots, u_\ell$ occurring in the target t such that u_{i+1} occurs to the right of u_i in t and for any given $i \in \{1..l-1\}$, u_i and u_{i+1} are similar enough, i.e., they differ by at most k editing operations. The editing operations considered in this paper are insertions, deletions and substitutions - see Fig. 1.

Crochemore et al. (1998) presented an algorithm for computing non-overlapping evolutionary chains in monophonic targets. Their algorithm runs in time $O(nm)$, where n and m denote the length of the target and the motif, respectively. They also presented an $O(n(\log m + \log |\Sigma|))$ theoretical version algorithm for the same problem that makes use of suffix trees and another version that requires $O(kn)$ time for fixed alphabets. Here Σ and k denote the size of the underlying alphabet and the approximation threshold, respectively. Furthermore, they considered also several variants of overlapping evolutionary chains for which they presented $O(n^2)$ algorithms.

Such algorithms are of great use in music analyses, for musical motifs may actually evolve this way. One actual case is shown by the successive thematic entries present in Messiaen's piano work, *Vingt Regards sur L'Enfant Jesus*. Other simple examples are the familiar cases of the standard tonal answer in a conventional fugue, or the increasingly elaborated varied reprises of an 18th-century rondo theme. On a more subtle level, the *idée fixe* in Berlioz's *Symphonie Fantastique* recurs in a wide variety of different forms throughout the four movements of the symphony. In all these cases, each repetition can be seen as a transformation of the original motif. However, in these cases a repetition of generation r is often more similar to the repetition of generation $r-1$ than to the original motif; a measure of this similarity has to be preset in an algorithm intended to detect all such repetitions of the motif.

In practice, the music under consideration is usually polyphonic (as in the examples above) in which case the algorithms by Crochemore et al. (1998) cannot be applied. However, when dealing with polyphony, problems become more challenging. For instance, for traditional

pattern matching problem in music numerous algorithms for the monophonic case have been suggested (see e.g. (Ghias et al., 1995; Lemström and Laine, 1998; McNab et al., 1997; Pollastri, 1999; Rolland et al., 1999; Shmulevich et al., 2001)) but only a few for the polyphonic case (Holub et al., 1999; Lemström and Tarhio, 2000; Meredith et al., 2001). The problem of *distributed pattern matching in polyphonic musical sequences* is as follows: given a set of sequences of notes (one sequence for each voice) and a pattern, find whether occurrences of the pattern occur distributed across the sequences. More formally, given a set t of strings $t^i = t_1^i \dots t_n^i$, $i \in \{1..h\}$, for some constant h and a pattern $p = p_1 \dots p_m$, we say that p occurs at position j of t if $p_1 = t_j^{i_1}, p_2 = t_{j+1}^{i_2}, \dots, p_m = t_{j+m-1}^{i_m}$ for some $\{i_1 \dots i_m\} \in \{1..h\}$. Although the techniques solving this problem (Holub et al., 1999; Lemström and Tarhio, 2000) may also be adapted to approximate matching, they are not applicable to the problem considered here.

Our problem of *motif evolution in polyphonic music* is as follows: given a set t of strings $t^i = t_1^i \dots t_n^i$, $i \in \{1..h\}$, for some constant h ³ and a motif p , find whether there exists a sequence $u_1 = p, u_2, \dots, u_\ell$ occurring in the target t such that u_{j+1} occurs to the right of u_j in t and for any given $j \in \{1..l-1\}$ u_j and u_{j+1} are k -similar (i.e. they differ by at most k insertions, deletions and substitutions). Moreover, every u_j occurs within one voice of t (any pair (u_j, u_k) , however, may occur in distinct voices).

This paper is organized as follows. The next Section presents basic definitions for strings and background notions for string pattern-matching. Section 3 and 4 show how *evolution trees* representing non-overlapping and overlapping chains, respectively, can be computed. In Section 5 we show how the solutions to the three variants of our problem, that are *the longest evolutionary chain*, *the nearest neighbour evolutionary chain*, and *the minimal weight chain*, can be induced out of the evolution trees. Finally, Section 6 presents conclusions and some open problems.

2 String Combinatorics

A *string* is a sequence of zero or more symbols from an alphabet Σ ; the “no symbol”, that is, the string with zero symbols, is denoted by ϵ . The set of all strings over the

¹In monophonic music, there is only one note played at a time.

²In polyphonic music, at times, several notes are played simultaneously.

³Note that $h = 1$ represents the degenerated monophonic case.

	1	2	3	4	5	6	7	8
ρ	B	A	D	F	E	ϵ	C	A
τ	B	C	D	ϵ	E	F	C	A
	Mat	Mis	Mat	Del	Mat	Ins	Mat	Mat

Figure 2: Types of differences: Mismatch, Insertion, Deletion.

alphabet Σ is denoted by Σ^* . A string x of length n ($|x| = n$, for short) is represented by $x_1 \dots x_n$, where $x_i \in \Sigma$ for $1 \leq i \leq n$. A string w is a *substring* of x if $x = uwv$ for $u, v \in \Sigma^*$; we equivalently say that the string w occurs at position $|u| + 1$ of the string x . The position $|u| + 1$ is said to be the *starting position* of w in x and the position $|u| + |w|$ the *ending position* of w in x . A string w is a *prefix* of x if $x = wu$ for $u \in \Sigma^*$. Similarly, w is a *suffix* of x if $x = uw$ for $u \in \Sigma^*$.

Consider two sequences $\tau = \tau_1 \tau_2 \dots \tau_r$ and $\rho = \rho_1 \rho_2 \dots \rho_r$ with $\tau_i, \rho_i \in \Sigma, i \in \{1 \dots r\}$. If $\tau_i = \rho_i$, then we say that τ_i and ρ_i *match*, otherwise τ_i *differs* from ρ_i . We distinguish among the following three types of differences:

1. Neither of these two symbols correspond to “no symbol” and the symbol of the first sequence corresponds to a different symbol of the second one, that is $\tau_i \neq \rho_i, \tau_i \neq \epsilon$ and $\rho_i \neq \epsilon$. This type of difference is a *mismatch* (or a substitution).
2. The symbol of the first sequence corresponds to “no symbol” of the second sequence, i.e., $\rho_i \neq \epsilon$ and $\tau_i = \epsilon$. This type of difference is called a *deletion*.
3. The symbol of the second sequence corresponds to “no symbol” of the first sequence, that is $\rho_i = \epsilon$ and $\tau_i \neq \epsilon$. This type of difference is called an *insertion*.

To give an example, let $\rho = BADFECA$ and $\tau = BCDEFCA$ (see Fig. 2). If we consider the alignment above, matches occur at positions 1, 3, 5, 7 and 8, while there is a mismatch at position 2, a deletion at position 4 and an insertion at position 6. Another way of seeing this difference is that one can transform the sequence ρ to τ by using the basic operations: insertions, deletions and substitutions. In this example, it means that we need three basic operations to transform $BADFECA$ into $BCDEFCA$: one mismatch (position 2 (A, C)), one deletion (position 4 (F, ϵ)) and one insertion (position 6 (ϵ, F)).

Note that we can also use three substitutions (position 2 (A, C), position 4 (F, E) and position 5 (E, F), see Fig. 3) without using the insertion and deletion. Therefore an “optimal alignment” is not necessarily unique. Nevertheless, we can always compute the minimal number of operations to transform one string into the other. If ρ is obtainable from τ (or vice versa) by using k editing oper-

	1	2	3	4	5	6	7
τ	B	A	D	F	E	C	A
ρ	B	C	D	E	F	C	A
	Mat	Mis	Mat	Mis	Mis	Mat	Mat

Figure 3: An alternative solution.

3 Computing Evolution Tree for Non-Overlapping Chains

Let p be a motif to be searched for in a polyphonic target t of h parallel voices. We aim at finding whether there exists a chain of monophonic strings $\mathcal{U} = \{u_1 = p, u_2, \dots, u_\ell\}$ occurring in the target.

Consider an occurrence of u_i in t . Function $ind(u_i)$ shows the index (subscript) of the matching position of u_i in t . For example, let $h = 1, t = \text{combinatorics}$, and $u_2 = \text{tori}$. Then $ind(u_{23}) = 10$. A chain of monophonic strings $\mathcal{U} = \{u_1, \dots, u_\ell\}$ is called *Non-Overlapping Evolutionary Chain (NOEC)* if and only if it satisfies the conditions:

1. $u_1 = p$;
2. each u_i occurs in some $t^{k_i}, i \in \{1 \dots \ell\}, k \in \{1 \dots h\}$;
3. $ind(u_{i+1}) > ind(u_{i_{|u_i|}})$, for $1 \leq i < \ell$; and
4. $u_i =_k u_{i+1}$, for $1 \leq i < \ell$.

Our idea is to compute an evolution tree representing all non-overlapping evolutionary chains of the motif. In Section 5, we show how solution to different variants of the problem can be induced out of the tree. The main idea of the algorithm is to maintain the scores of differences between every prefix of the motif and the target. We keep on calculating the scores until we found an occurrence of p in a voice of t with at most k -differences. Once an occurrence of the motif has been found at position j on the target, the match would be extracted out of the target and this match will be used in a new query that starts at position $j + 1$.

The algorithm presented below uses the notion of recursion to compute NOECs.

1. Initialization:

2. Main step:

At step j we have computed the number of differences between $t_1^i \dots t_j^i, 1 \leq j \leq n, i \in \{1 \dots h\}$ and p .

3. Check for match

If the number of differences between p and $t_1^i \dots t_j^i$ is at most k for some pair (i, j) , then let u be the suffix of that prefix having k differences with p .

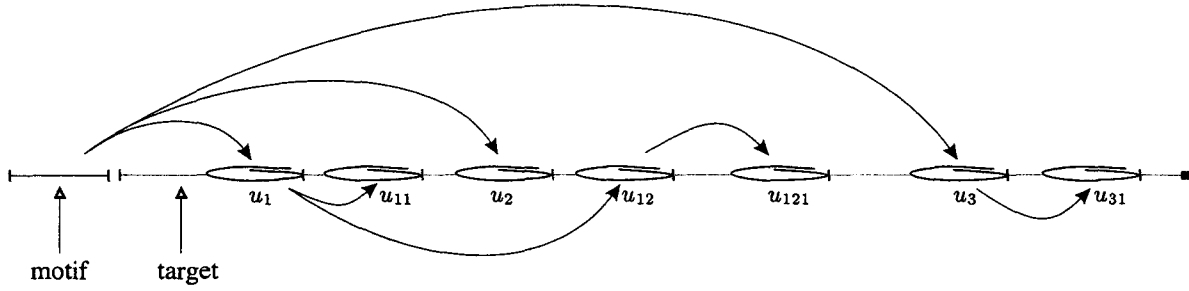


Figure 4: The original motif and how it evolves.

an occurrence of u with at most k -differences in $t_{j+1} \dots t_n$.

4. Output chain

If u is a leaf of the tree (see Fig. 5) then output the current chain and continue the previous recursive call.

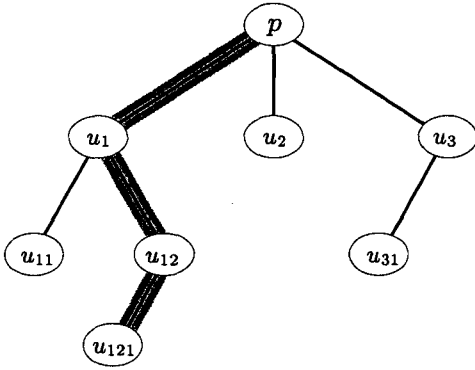


Figure 5: The tree representation. The longest chain is emphasized by shading.

Fig. 4 shows how the algorithm above works. The first match with the original motif is u_1 and this becomes the next motif. The position of the match becomes the starting position for the next recursive call. On continuing the search, u_1 is matched with u_{11} and u_{12} . However, any occurrences for u_{11} is not found in t , hence $\{p \rightsquigarrow u_1 \rightsquigarrow u_{11}\}$ is a chain, so is $\{p \rightsquigarrow u_1 \rightsquigarrow u_{12} \rightsquigarrow u_{121}\}$. Once the search for u_1 further in the target has been exhausted, the next match with the original motif is taken, i.e., u_2 . The same procedure is repeated and the following chains are found: $\{p \rightsquigarrow u_2\}$, $\{p \rightsquigarrow u_3 \rightsquigarrow u_{31}\}$.

The same figure can be rearranged as a tree (see Fig. 5) which gives a more accurate representation of how the Evolution Tree is produced. All our discussions will be based on this tree representation and a leaf of this tree always completes a chain.

3.1 Pseudo-Code

Let us consider the straightforward NOEC's main routine (see Fig. 6).

```

NOEC( $t, p, k$ )
1   $EvoChain \leftarrow \{\emptyset\}$ 
2   $ETREE(1, p, EvoChain)$ 

```

Figure 6: Algorithm for computing non-overlapping evolutionary chains.

The input for NOEC are target (t), motif (p) and error tolerance (k). To produce the evolution tree, NOEC calls ETREE (the first parameter gives the starting position for the search).

```

ETREE( $start, p, EvoChain$ )
1  if  $start > n - \lfloor m/2 \rfloor$ 
2  then PRINT( $EvoChain$ )
3  return
4   $IsLeaf \leftarrow TRUE$ 
5  for  $j \leftarrow start$  to  $n$ 
6  do if  $D(p, t_{start..j}) \leq k$ 
7  then  $IsLeaf \leftarrow FALSE$ 
8          $p' \leftarrow BACKTRACK(p, t_{start..j})$ 
9         ▷ the recursive call
10         $ETREE(j + 1, p', EvoChain \cup \{start\})$ 
11  if ( $IsLeaf$ )
12  then PRINT( $EvoChain$ )

```

Figure 7: Recursive function that computes the evolution tree.

Consider the pseudo-code for the function ETREE that is given in Fig. 7. At line 9, the all important recursive call with the new motif p' can be found. At line 8, BACKTRACK is the process of retrieving the match out of the

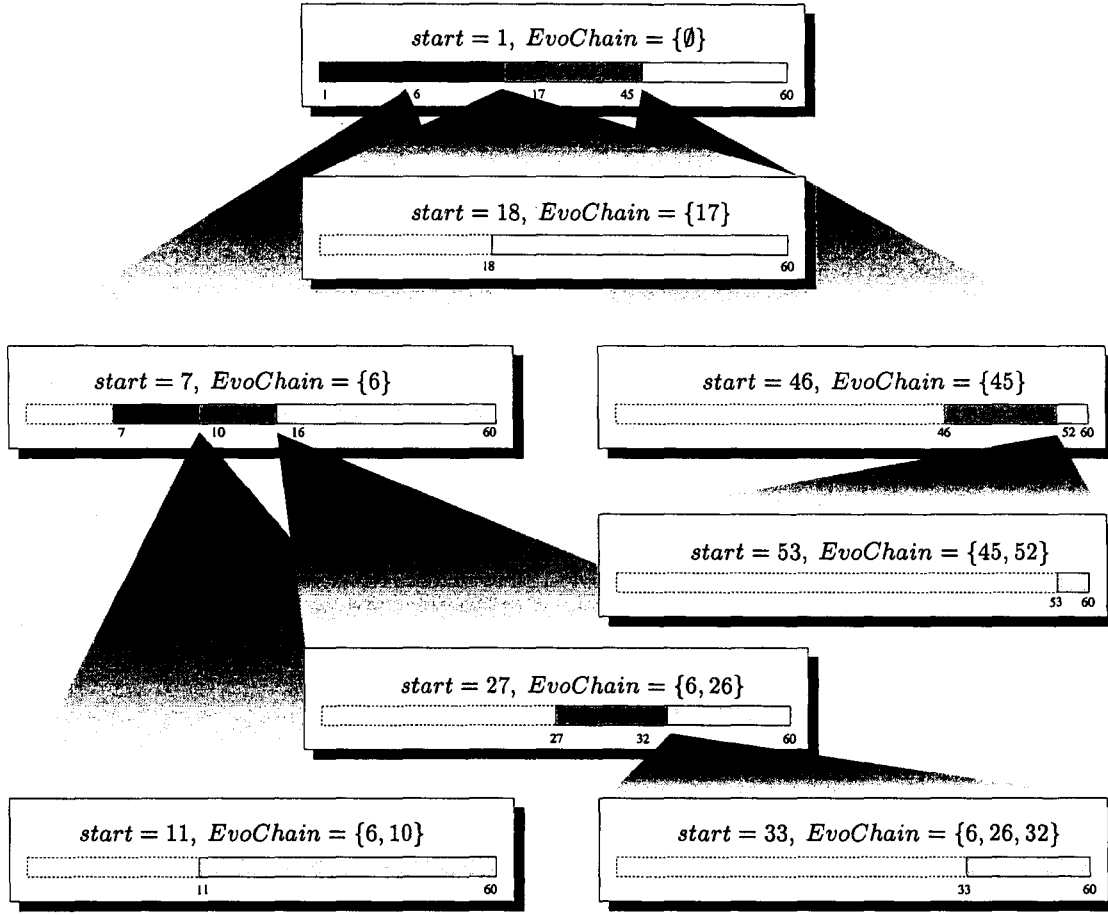


Figure 8: Example showing how the recursive calls from the tree.

possible to find a match after position $n - \lfloor m/2 \rfloor$. Thus, we have reached a leaf, i.e., end of a chain. Similarly line 11 outputs the chain if a match is not found in the current iteration, again this is a leaf of the evolution tree.

Referring to the tree representation shown in Fig. 5, assume $u_1 = 6$, $u_2 = 17$, $u_3 = 45$, $u_{11} = 10$, $u_{12} = 26$, $u_{31} = 52$ and $u_{121} = 32$ with $n = 60$. Fig. 8 shows how the function ETREE recursively computes the chains.

Table 1 is the tracing of the algorithm NOEC. The first entry in the table is the initial call to ETREE from algorithm NOEC. Entry 1 indicates that the first recursive call after a match (with at most k -differences) was found ending at position 6. Therefore, 6 was added to the *EvoChain* and the new task is to find “ABCD” starting from position 7. This makes sure the matches do not overlap. Entry 2 denotes that a k -similar match was found ending at position 10, which is then added to the chain. Now the new recursive query is to find “ABDD” starting from position 11. This query, however, does not yield any results. Therefore, we must have reached a leaf on the evolution tree (see Fig. 5), and the evolutionary chain *EvoChain* is output. Having finished with this instance of ETREE, we go back to entry 1 and continue from posi-

tion 6 (which is where we left and spawned entry 2) with the old motif “ACD” until another k -similar match is found. Ending at position 26 is an approximate match which is added to *EvoChain*. Now ETREE is recursively called to look for “ACCB” starting from position 27 as depicted by entry 3 on the table. This process of recursively calling ETREE with a new query each time is repeated until the first call to ETREE (entry *init* in Table 1) is exhausted. Table 1 is the tracing of the recursive calls.

3.2 Running Time

The computation of a single non-overlapping evolutionary chain requires $O(mn)$ time and $O(mn)$ space. The computation of all non-overlapping evolutionary chains can require exponential time (in terms of nm) in pathological worst-case scenarios like this one:

$$p = A^m, t = A^n \text{ and } k = 0. \quad (1)$$

The worst case happens, when we have a match at every position. The best case happens when there is no match at all; In this case $O(mn)$ time and $O(m)$ space is

call	start	EvoChain	p
init	1	$\{\emptyset\}$	"AACD"
1	7	$\{6\}$	ABCD
2	11	$\{6, 10\}$	ABDD
Output EvoChain $\leftrightarrow \{ABCD \rightsquigarrow ABDD\}$			
3	27	$\{6, 26\}$	ACCB
4	33	$\{6, 26, 32\}$	BCCB
Output EvoChain $\leftrightarrow \{ABCD \rightsquigarrow ACCB \rightsquigarrow BCCB\}$			
5	18	$\{17\}$	BACD
Output EvoChain $\leftrightarrow \{BACD\}$			
6	46	$\{45\}$	AABD
7	53	$\{45, 52\}$	AABB
Output EvoChain $\leftrightarrow \{AABD \rightsquigarrow AABB\}$			

Table 1: Recursion parameters and order.

algorithm for computing all non-overlapping evolutionary chains is quadratic $O((mn)^2)$, requiring $O(mn)$ space.

A practical speed-up for the algorithm is to mark the nodes that have already been used in a chain, and only unmarked nodes can be selected as nodes in any chain. In this way, even the exponential worst case time complexity of the pathologic case becomes polynomial: $O(n^4)$.

4 Computing Evolution Tree for Overlapping Chains

In this section we present the other variation of evolutionary chains, namely Overlapping Evolutionary Chains (OEC, for short). The problem is defined as follows: given a set t of strings $t^i = t_1^i \dots t_n^i$, $i \in \{1..h\}$, a motif p and an integer $k < |p|/2$, find whether there exists a sequence $u_1 = p, u_2, \dots, u_\ell$ occurring in the target t such that the following conditions are satisfied (note the difference between the items 3 here and of that given in Section 3):

1. $u_1 = p$;
2. each u_i occurs in some t^{k_i} , $i \in \{1 \dots \ell\}$, $k \in \{1 \dots h\}$;
3. $\text{ind}(u_{i+1}) > \lceil \text{ind}(u_i) + \frac{|u_i|}{2} \rceil$, for $1 \leq i < \ell$;
4. $u_i =_k u_{i+1}$, for $1 \leq i < \ell$.

These strings have been constrained to overlap at most $|p|/2$ symbols. Without such a constraint, we can obtain trivial chains such as $\text{ind}(u_i) = s$, $\text{ind}(u_{i+1}) = s + 1$, and obviously u_i and u_{i+1} have at most one difference.

Let us now introduce the pseudo-code for the OEC algorithm (see Fig. 9). The function ETREE' (Fig. 10) is a slightly modified version of the one that was used with

```

OEC( $t, p, k$ )
1   $\text{EvoChain} \leftarrow \{\emptyset\}$ 
2   $\text{ETREE}'(1, p, \text{EvoChain})$ 

```

Figure 9: Algorithm for computing overlapping evolutionary chains.

Note that the two versions, ETREE and ETREE' , are very similar: the only difference is the addition of the three lines at the beginning. Line 1 makes sure the overlapping cases are taken into account but with the constraint ($m/2$) as explained earlier. The if statement in Line 2 is for the case when the initial call from NOEC is met with $\text{start} = 1$ and having executed Line 1, start will be negative (thus, start is fixed to be positive).

Obviously, the time complexity of this algorithm is of the same order as that of NOEC. Since we allow for overlapping, $m/2$ symbols are included again in each recursive call. More precisely, the practical time complexity for OEC is $O((mn)^2)$ and space complexity $O(mn)$.

5 Inducing Solutions out of the Evolution Trees

Let us now define three specific problems of evolutionary chain computing and show how solutions for these problems can be induced out of the evolution trees.

Longest Evolutionary Chain (LEC). LEC is the simplest form of these problems; it is the chain $\mathcal{U} = u_1 \dots u_\ell$ that maximizes ℓ . Once the evolution tree has been computed, the length of the LEC is the height of the evolution tree. Therefore every time a new chain is being output,

```

ETREE'(start, p, EvoChain)
1  start ← start - m/2
2  if start < 1
3    then start ← 1
4  if start > n - ⌊m/2⌋
5    then PRINT(EvoChain)
6    return
7  IsLeaf ← TRUE
8  for j ← start to n
9    do if D(p, tstart..j) ≤ k
10     then IsLeaf ← FALSE
11         p' ← BACKTRACK(p, tstart..j)
12         ▷ the recursive call
13         ETREE'(j + 1, p', EvoChain ∪ {start})
14  if (IsLeaf)
15    then PRINT(EvoChain)

```

Figure 10: Recursive function to compute the evolution tree for OEC.

the length of that chain is compared with the height of the tree, and if it longer then that must be the longest chain so far. This procedure is repeated for all the chains and in the end we output the length of the longest chain together with the indices of the chain elements u_i found in the target. In the example given in Fig. 5 LEC is emphasized using shading.

Nearest neighbour Evolutionary Chain (NEC). NEC is the chain $\mathcal{U} = \{u_1, \dots, u_\ell\}$ that minimizes e in the following equation:

$$e = \sum_{i=1}^{\ell-1} f(\text{ind}(u_{i+1}) - \text{ind}(u_{i|u_i|})), \quad (2)$$

where f is some increasing function on positive integers. The simplest function of that form (that is also considered here) is the identity function; $f(x) = x$.

When solving NEC, we associate values with the edges of the tree. These values are the gaps between the two joining nodes of the tree. In the case of overlapping matches, the gap is taken to be zero. Once this is completed, the total gap of each chain is compared to find the minimum chain and the total gap together with the chain index is output.

Minimal weight Evolutionary Chain (MEC). MEC minimizes d in:

$$d = \sum_{i=1}^{\ell-1} D(u_i, u_{i+1}), \quad (3)$$

where $D(u_i, u_{i+1})$ is the Edit Distance between u_i and

In this case, we attach values to the edges, as well. This time, the values are the the number of differences, d' , between the two joining nodes of the tree. Once this is completed, the total difference d of each chain is summed up and compared to find the chain with minimum d . The output is the chain giving the minimum to d and the value d , itself.

6 Conclusions and Open problems

Our primary goal is to identify efficient algorithms for computational problems which arise in computer-assisted analysis of music, and to also formalize their relation to well known string pattern-matching problems. The primary direction of this research is towards a formal definition of musical similarity between musical entities (i.e. complete pieces of music or meaningful subsets of pieces, e.g. 'themes' or 'motifs', see (Cambouropoulos, 1997; Cambouropoulos and Smaill, 1995; Crawford et al., 1998; Lemström, 2000) for details). In particular we are aiming at producing a quantitative measure or 'characteristic signature' of a musical entity. This measure is essential for melodic recognition and it will have many uses including, for example, data retrieval from musical databases.

We presented practical algorithms, NOEC and OEC, for computing non-overlapping and overlapping evolutionary chains. Furthermore, we presented three variants of these problem, the longest evolutionary chain, the nearest neighbour evolutionary chain, and the minimal weight evolutionary chain, each of which are of practical importance.

The problems presented here need to be further investigated under a variety of *similarity* or *distance* rules (see (Crawford et al., 1998; Mongeau and Sankoff, 1990; Lemström, 2000)). For example, *Hamming distance* of two strings u and v is defined to be the number of substitutions necessary to get u from v (u and v have the same length). Several variants to the evolutionary chain problem are still open. The choice of suitable similarity criteria in music and biology is still under investigation. The use of penalty tables may be more suitable than the k -differences criterion in certain applications.

Further investigation whether methods such as (Galil and Park, 1990; Landau and Vishkin, 1988) can be adapted to solve the problems considered here is needed. Further-

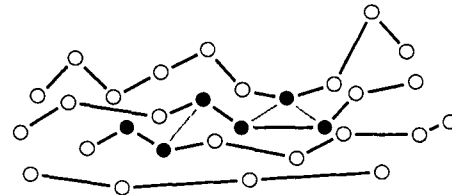


Figure 11: Motif distributed across voices.

more, modifications to our algorithms in order to find evolutionary occurrences that are distributed across voices (see Fig. 11) are left to future studies; the algorithms that have been presented to find distributed occurrences of a monophonic motif in a polyphonic target (Holub et al., 1999; Lemström and Tarhio, 2000) are not applicable to the problem at hand. This is because these bit-parallel algorithms are only able to locate the positions of occurrences, but they cannot, however, extract the matching substring out of the target. Therefore, they cannot be applied in a recursive manner as the algorithms presented here.

Acknowledgements

Costas Iliopoulos was partially supported by a Marie Curie Fellowship and Royal Society, Wellcome Foundation and NATO grants. Kjell Lemström was partially supported by the grants #48313 from the Academy of Finland and research grant GR/R25316 from EPSRC. Jose Pinzón was partially supported by an ORS studentship and EPSRC Project GR/L92150.

References

- A. Apostolico. The myriad virtues of the suffix trees. *Theoretical Computer Science*, 22:297–315, 1983.
- O. Berkman, C. Iliopoulos, and K. Park. String covering. *Information and Computation*, 123:127–137, 1996.
- E. Cambouropoulos. A formal theory for the discovery of local boundaries in a melodic surface. In *Proceedings of the III Journées d'Informatique Musicale*, Caen, France, 1996a.
- E. Cambouropoulos. A general pitch interval representation: Theory and applications. *Journal of New Music Research*, 25:231–251, 1996b.
- E. Cambouropoulos. The role of similarity in categorisation: Music as a case study. In *Proceedings of the Third Triennial Conference of the European Society for the Cognitive Sciences of Music (ESCOM)*, Uppsala, 1997.
- E. Cambouropoulos and A. Smaill. A computational theory for the discovery of parallel melodic passages. In *Proceedings of the XI Colloquio di Informatica Musicale*, Bologna, Italy, 1995.
- T. Crawford, C.S. Iliopoulos, and R. Raman. String matching techniques for musical similarity and melodic recognition. *Computing in Musicology*, 11: 71–100, 1998.
- M. Crochemore, C.S. Iliopoulos, and H. Yu. Algorithms for computing evolutionary chains in molecular and musical sequences. In *Proceedings of the 9-th Australasian Workshop on Combinatorial Algorithms*, volume 6, pages 172–185, 1998.
- M. Crochemore and W. Rytter. *Text Algorithms*. Oxford University Press, 1994.
- V. Fischetti, G. Landau, J. Schmidt, and P. Sellers. Identifying periodic occurrences of a template with applications to protein structure. In *Proc. 3rd CPM*, volume 644, pages 111–120. Lecture Notes in Computer Science, 1992.
- Z. Galil and K. Park. An improved algorithm for approximate string matching. *SIAM Journal on Computing*, 19:989–999, 1990.
- A. Ghias, J. Logan, D. Chamberlin, and B.C. Smith. Query by humming - musical information retrieval in an audio database. In *ACM Multimedia 95 Proceedings*, pages 231–236, San Francisco, CA, 1995.
- D. Gusfield. *Algorithms on strings, trees and sequences: computer science and computational biology*. Cambridge University Press, Cambridge, 1997.
- J. Holub, C.S. Iliopoulos, B. Melichar, and L. Mouchard. Distributed string matching using finite automata. In *Proceedings of the 10th Australasian Workshop On Combinatorial Algorithms*, pages 114–128, Perth, 1999.
- C.S. Iliopoulos, D.W.G. Moore, and K. Park. Covering a string. *Algorithmica*, 16:288–297, 1996a.
- C.S. Iliopoulos, D.W.G. Moore, and W.F. Smyth. A linear algorithm for computing the squares of a fibonacci string. In *Proceedings CATS'96, Computing: Australasian Theory Symposium*, pages 55–63, 1996b.
- C.S. Iliopoulos and L. Mouchard. An $o(n \log n)$ algorithm for computing all maximal quasiperiodicities in strings. In *Proceedings of CATS'99: Computing: Australasian Theory Symposium*, volume 21, pages 262–272, Auckland, New Zealand, 1999. Lecture Notes in Computer Science.
- S. Karlin, M. Morris, G. Ghandour, and M.Y. Leung. Efficient algorithms for molecular sequences analysis. In *Proc. Natl. Acad. Sci.*, volume 85, pages 841–845, 1988.
- G.M. Landau and J.P. Schmidt. An algorithm for approximate tandem repeats. In *Proc. Fourth Symposium on Combinatorial Pattern Matching*, volume 648, pages 120–133. Lecture Notes in Computer Science, 1993.
- G.M. Landau and U. Vishkin. Fast string matching with k differences. *Journal of Computer and Systems Sciences*, 37:63–78, 1988.

- K. Lemström. *String Matching Techniques for Music Retrieval*. PhD thesis, University of Helsinki, Department of Computer Science, 2000. Report A-2000-4.
- K. Lemström and P. Laine. Musical information retrieval using musical parameters. In *Proceedings of the 1998 International Computer Music Conference*, pages 341–348, Ann Arbor, MI, 1998.
- K. Lemström and J. Tarhio. Detecting monophonic patterns within polyphonic sources. In *Content-Based Multimedia Information Access Conference Proceedings (RIAO'2000)*, volume 2, pages 1261–1279, Paris, 2000.
- G. Main and R. Lorentz. An $O(n \log n)$ algorithm for finding all repetitions in a string. *Journal of Algorithms*, 5: 422–432, 1984.
- R.J. McNab, L.A. Smith, D. Bainbridge, and I.H. Witten. The New Zealand digital library MELody inDEX. *D-Lib Magazine*, 1997.
- D. Meredith, G.A. Wiggins, and K. Lemström. Pattern induction and matching in polyphonic music and other multi-dimensional datasets. In *the 5th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI'2001)*, volume X, pages 61–66, Orlando, FLO, July 2001.
- A. Milosavljevic and J. Jurka. Discovering simple dna sequences by the algorithmic significance method. *Comput. Appl. Biosci.*, 9:407–411, 1993.
- M. Mongeau and D. Sankoff. Comparison of musical sequences. *Computers and the Humanities*, 24:161–175, 1990.
- D.W.G. Moore and W.F. Smyth. Computing the covers of a string in linear time. In *Proc. 5th ACM-SIAM Symposium on Discrete Algorithms*, pages 511–515, 1994.
- E. Myers and S. Kannan. An algorithm for locating non-overlapping regions of maximum alignment score. In *Proc. Fourth Symposium on Combinatorial Pattern Matching*, volume 648. Lecture Notes in Computer Science, 1993.
- P.A. Pevzner and W. Feldman. Gray code masks for dna sequencing by hybridization. *Genomics*, 23:233–235, 1993.
- E. Pollastri. Melody-retrieval based on pitch-tracking and string-matching methods. In *Proceedings of the XIIth Colloquium on Musical Informatics*, 1999.
- P.Y. Rolland, G. Raskinis, and J.G. Ganascia. Musical content-based retrieval: an overview of the melodiscov approach and system. In *ACM Multimedia 99 Proceedings*, Orlando, FLO, 1999.
- J.P. Schmidt. All shortest paths in weighted grid graphs and its application to finding all approximate repeats in strings. In *Proc. of the Fifth Symposium on Combinatorial Pattern Matching CPM'94*. Lecture Notes in Computer Science, 1994.
- I. Shmulevich, O. Yli-Harja, E. Coyle, D.J. Povel, and K. Lemström. Perceptual issues in music pattern recognition - complexity of rhythm and key finding. *Computers and the Humanities*, 35(1):23–35, 2001.
- S.S. Skiena and G. Sundaram. Reconstructing strings from substrings. *J. Computational Biol.*, 2:333–353, 1995.
- D.A. Stech. A computer-assisted approach to micro-analysis of melodic lines. *Computers and the Humanities*, 15:211–221, 1981.

III – Creative Language and Context Generation

Linguistic Creativity at Different Levels of Decision in Sentence Production

Pablo Gervás

Universidad Complutense de Madrid
Ciudad Universitaria, 28040 Madrid, Spain
pgervas@sip.ucm.es

Abstract

The shape taken by linguistic creativity at the different levels of decision involved in sentence production (phonetics, rhythm, lexical choice, semantics, syntax and narrative content) is explored in relation to existing computational models of creativity. A general outline of the possibilities is given for each level, and two specific levels - word invention at the lexical level, illustrated by the Jabberwocky poem by Lewis Carroll; and poetic metaphor at the semantic level, illustrated by examples from verses by Garcia Lorca - are studied in further detail. The applicability of the existing computational models is discussed in connection to the kind of creativity apparent in the examples.

1 Introduction

Assuming we all speak a common language, everybody uses very much the same grammar and the same words to build the sentences that make up our daily world. How come some of these sentences are considered creative and some are not?

One way to begin to answer this question is to identify the levels of decision at which the final form of a sentence is shaped. Some of these allow little variation (grammar or syntax), others provide a big field (semantics), others are only immediately available to the trained speaker (rhythm, prosody), others are restricted to use by the poet (alliteration). A complete study of all the possibilities would require an enormous amount of space. As a first approximation, six basic levels of decision can be identified in the production of linguistic elements:

- phonetics, the level at which letters are put together to make sounds
- rhythm, the level at which the stress patterns of words are taken into account to shape the stress pattern of a sentence or a text
- lexical choice, the level at which actual words are chosen for the text
- semantics, the level at which the meanings of the words being used are considered and put together to form the meaning of the text
- syntax, the level at which the linguistic constructions used to join the words (and their meanings) to one another are chosen

- narrative, the level at which the contents of the text are decided

Another possible source of insights is to consider what objectives drive the production of sentences. Sentences are produced in many different contexts, and with many different purposes. Decisions taken at these levels on the final shape of a sentence will necessarily take into account a number of objectives of the speaker/writer. In most instances of language generation, the objectives that drive the utterance process are of a practical nature, related with the communication of a certain message or information. In these cases, the narrative and the semantic levels take priority over all the remaining levels, and transgressions - sometimes dramatically severe - of the accepted elementary rules governing language production are allowed. A speaker in a hurry may, for example, waive the rules of correct syntax as long as he sees his message put across briefly. In going beyond the accepted rules, such a speaker may be deemed to be behaving creatively. This type of linguistic creativity (say, corner-cutting creative communication) is worth exploring in detail, but it would require access to enough samples of specific instances of the phenomenon to provide starting material. Other instances of language generation, have objectives specifically geared towards obtaining a pleasing effect of some sort. These instances tend to get explicitly recorded for this pleasing effect to be available at later times, and they provide an easier starting point for a study of this sort. To make matters even easier, this study will concentrate on literary written texts, even though there are many other fields and formats in which there can be said to be a con-

scious linguistic creative effort with an aesthetic aim in mind (film dialogues, TV scripts, radio programs, advertising...).

Finally the point of view from which the question is asked plays a role. When considering whether a sentence is creative or not, it seems important to take into account three basic issues: whether the speaker or writer considers he has been creative in producing that sentence, whether other people consider the sentence creative, and whether the sentence can be considered a valid sentence of the language. These three basic issues already begin to reflect the elementary distinctions outlined by Ritchie (2001) in his sketch of creativity assessment. The matter is considered in more depth in the following sections.

Creativity at the different levels of decision is outlined in this section, formal concepts of creativity are discussed in section 2, two specific cases are developed further in sections 3 and 4, and conclusions are drawn in section 5.

1.1 Phonetics

At the level of phonetics, linguistic creativity may be aimed at searching for a pleasing aesthetic effect by playing with a careful selection of the words in the sentence with an eye on the effect that results over the phonetics of the complete sentence. The results of this type of creativity come up as pleasing uses of rhyme, internal rhyme or alliteration (Espy, 1997). The extreme example is that of sound poetry (Ball, 1974; Hausman, 1971; Schwitters, 1993), an artistic initiative related to the Dada and Surrealist movement. In sound poetry, poems are not built up using words but simple phonetic constructs without meaning. The classic example of this line of creative work is Schwitters' *Ursonate*, a forty minute long phonetic poem set in a more or less sonata form. This particular approach to composition survives in text-sound composition, an artistic hybrid standing midway between poetry and music. For a review of text-sound composition efforts see Hultberg (1993).

1.2 Rhythm

One can find more creative rhythms that are uncommon in the language, or in the existing poetry. Edgar Allan Poe, in his defence of this poem the *Black Raven* (Poe, 1997) specifically discusses how the rhythm he has chosen for his verses is innovative - in the sense that it had not been used before to his knowledge, in English literature.

1.3 Lexical

The choice of vocabulary with which to construct a sentence plays an important role in making the result pleasing, but one is usually restricted to words that the reader will understand, leaving little room for creativity. A

different alternative lies in using words that the user does not know. This forces the rules to a certain extent, and, if done carefully, it can be done in such a way as to actually convey a certain meaning to the reader in spite of the unknown words. A good example of this is the poem 'Jabberwocky', that appears in Alice's adventures through the Looking Glass (Carroll, 1872). This particular poem is analysed later in the paper.

1.4 Semantics

If we approximate the pictures in our head by means of a logical description of them (a challengeable assumption), a rough and ready formalization of what it takes for a sentence to be semantically creative is to consider as a measure of creativity the ratio between the size of the sentence and the complexity of the picture (a short sentence that manages to create a complex picture may suggest special creativity). However, this may be more related to the author's craftsmanship. Another possibility lies in forcing the semantics a little bit. Truly creative sentences seem to put together words that force an interpretation where, in merging the meanings of the words, the reader must necessarily prune one or the other to reach the interpretation intended by the writer. This is particularly the case in the use of metaphor. The amount of pruning required (often triggered by wild clashes between the meanings of words purposefully joined together) gives an idea of the creativity involved. The degree of clashing can be a measure of the creativity involved. A set of examples of metaphors by Spanish poet Garcia Lorca is discussed in relation with this issue.

1.5 Syntax

Much may be said about literary creativity in the sphere of syntax, and most of it would require a precise statement of what non-creative syntax is taken to be, and therefore beyond the scope of this paper. There is much to be said in favour of a close study of the role of creativity in stretching syntax to achieve specific literary effects. This aspect is for the time being left in the hands of more able literary critics.

1.6 Narration

Beyond the boundary of a single sentence, there is a universe of creative possibilities to explore, related to the creativity employed in generating the situations that are being described or narrated. This is the level of creativity involved in general fiction, and the subject of study of thousands of academics world-wide. The implications of attempting to automate creativity at this level in terms of story telling systems have been tackled, among others, for animal stories (Meehan, 1977), Arthurian legend (Turner, 1992), humorous language (Ritchie, 2000) and

for tragic stories specifically concerned with treason (Bringsjord and Ferruci, 2000). Interesting as the field may be, the discussion in this paper focuses on creativity in the production of individual sentences in order to explore its role in language understanding and generation, and therefore this particular level is not explored in detail.

2. Formal accounts of creativity

Various attempts have been made to formalise the concepts involved in creativity in some more rigorous, formal way that allows objective judgements to be drawn.

2.1 Boden's framework

Boden's original framework (Boden, 1990) aims to describe AI approaches to creativity from a philosophical point of view. Two of the distinctions made in that work are particularly relevant to the issue in question. Boden distinguishes between *H-creativity* (creating a concept that has never been created at all) and *P-creativity* (creating a concept that has never been created before by a given creator). This distinction addresses the existence of a important subjective ingredient in our perception of creativity. She also distinguishes between *exploratory* creativity (identifying new concepts within a conceptual space that is already established) and *transformational* creativity (broadening an established conceptual space so that new concepts become accessible outside the bounds of the original conceptual space). Although this distinction opens the way for many interesting insights on creativity, it remains vague as to how an established conceptual space can be identified if not all of its members are known (a necessary requirement for exploratory creativity to be meaningful).

2.3 Extending Boden's framework

Wiggins (2001) outlines explicit definitions of Boden's concepts of exploratory and transformational creativity together with criteria for objectively distinguishing between them. The concepts developed here provide a framework in which the following two examples can be discussed.

Wiggins proposes a mechanism to describe an exploratory creative system in terms of a septuple:

$$\langle U, L, [[.]], \langle\langle . \rangle\rangle, R, T, E \rangle$$

where U represents a multidimensional space that includes all possible concepts, L is a language for expressing rules on members of U , R is a set of rules in L that defines a given conceptual space C included in U , T is a set of rules in L that encode the way in which a particular creative

step increments the set of known elements of a given conceptual space, and E is a set of rules for evaluating the quality of a concept. The function $[[.]]$ is an interpretation function which generates from L a function to select members of sets, and provides the mathematical tools to describe a conceptual space C in terms of the rules R that describe it, so that $C = [[R]](U)$. The function $\langle\langle . \rangle\rangle$ provides the means for defining the set of known concepts, c_o , after a given creative step in terms of the original set of known concepts, c_i , the sets of rules defining the conceptual space, R , and the creative means of traversal of the conceptual space, T ; so that $c_o = \langle\langle R \cup T \rangle\rangle(c_i)$. This characterization allows representation of several concepts that play an important role in an analysis of creativity: the rules that define a particular style (R), the rules that represent the *modus operandi* of a particular creator (T), or the consensus on what is a good concept for a given community at a given moment in time (E).

Wiggins argues that exploratory creativity operates by applying T to increment the set of known concepts. It is important to consider that, because creation takes time and effort and the amount of both devoted to exploring a conceptual space is limited in real terms – by the creator's life span at best – the set of concepts that may be reached using T within a given conceptual space may always be much bigger than the set of known concepts. This is what makes exploratory creativity interesting in practical terms, allowing timely discovery of specific concepts with high values for E .

Transformational creativity is characterised in terms of modifying either R – thereby adding new possible concepts – or T – providing means for discovering concepts that were possible but not accessible by previous means of creation. The intuition behind this is that the set of techniques available to a set of creators at a given moment in time may not be sufficient to traverse the complete conceptual space in which they are working.

An interesting idea that arises from the formalisation is that there is no formal constraint requiring that T be unable to lead to concepts not originally included in C – this is the reason why both R and T are needed to apply $\langle\langle . \rangle\rangle$ – thereby somewhat blurring the differences between exploratory and transformational creativity. In a way, this corresponds to accepting the fact that the creative technique of a given agent can be transformational in the sense that it leads beyond the initial conceptual space. Wiggins' proposal includes a conjecture that – the modification of R and T being itself a creative process – an equivalent formalisation can be used to describe transformational creativity, formulated in terms of creativity at

the meta-level. This implies that transformational creativity requires that the creator be aware of the particular methods that he is using in his work, so that he can be creative about them.

3 Inventing Words

The best known example of creativity at the lexical level is Carroll's poem 'Jabberwocky'. Its first few lines provide a fine sample of creativity in the use of invented words:

`Twas brillig, and the slithy toves
Did gyre and gimble in the wabe:
All mimsy were the borogoves,
And the mome raths outgrabe.

"Beware the Jabberwock, my son!
The jaws that bite, the claws that catch!
Beware the Jubjub bird, and shun
The frumious Bandersnatch!"

He took his vorpal sword in hand:
Long time the manxome foe he sought --
So rested he by the Tumtum tree,
And stood awhile in thought.

And, as in uffish thought he stood,
The Jabberwock, with eyes of flame,
Came whiffling through the tulgey wood,
And burbled as it came!

One, two! One, two! And through and through
The vorpal blade went snicker-snack!
He left it dead, and with its head
He went galumphing back.
"And, has thou slain the Jabberwock?
Come to my arms, my beamish boy!
O frabjous day! Callooh! Callay!"
He chortled in his joy.

`Twas brillig, and the slithy toves
Did gyre and gimble in the wabe;
All mimsy were the borogoves,
And the mome raths outgrabe.

3.1 A Close Look at the Poem

As Alice herself observes in the book, "Somehow it seems to fill my head with ideas -- only I don't exactly know what they are!". A very interesting analysis of complex processes involved in even the most unconscious interpretation of this piece of text is presented in (Dean, 2001).

Exactly how one may arrive at a computational model of the kind of creativity involved in generating this type of language samples is difficult to say. However, many clues are lying around to be gathered by the interested reader.

An important concept for this endeavour is provided in the original text (Carroll, 1872). Alice asks for an explanation of the poem, and Humpty Dumpty (with her help) indulges her whim. This explanation provides some insight into the processes that may be at play in composing the poem. For instance, the idea of a 'portmanteau' word - two meanings packed into one word - is explained. The examples given by Humpty Dumpty in his explanation of the poem are 'slithy', meaning 'lithe and slimy', or 'mimsy', meaning 'flimsy and miserable'.

Carroll himself goes further towards describing the actual process of creating a new word in a letter to Maud Standen in 1877 (Graham, 1981), discussing the word 'burble' that also occurs in the poem: "If you take the three verbs 'bleat', 'murmur' and 'warble', and select the bits I have underlined, it certainly makes 'burble'...". However, he presents us with an important difficulty in the rest of his sentence "...though I am afraid I can't distinctly remember having made it in that way." It should not be overlooked that, easy as it may seem to reconstruct in hindsight the way in which a poet arrived at a particularly creative word, this need not be the way in which it actually occurred to him.

An important issue is to identify how the poet ensures that a poem with an important amount of invented words can still be - at least partly - understood by the reader. Dean describes several techniques used by Carroll to keep 'Jabberwocky' from becoming complete nonsense:

- manufacture the words in such a way that they look as if they could be real (choose vowel and consonant combination that appear genuine and easily pronounceable),
- use many invented nouns and adjectives but comparatively few invented verbs,
- rely on the sound of the intended words (rather than their non-existent meaning) to convey the meaning of the poem,
- use the placement of the invented words within the sentences to give the reader an idea of how they function within the sentence.

From the point of view of computational accounts of creativity, a more revealing analysis is carried out by Hofstadter (1980) in terms of how the different translators of Jabberwocky chose to render the poem in different languages. Hofstadter considers the matter from the point of view of how the poem activates a symbolic network inside the brain of the reader. In particular, he studies how

translators devise versions of 'Jabberwocky' in a new language B, assuming that their aim is to find 'the same node' in the brain of the B reader as the one that activated by the original poem in the English reader. The main problem arises from the assumption that the symbolic network associated with the B language and English will usually be, on some level of analysis, extremely non-isomorphic: in a poem of this type many "words" do not carry ordinary meaning, but act purely as exciters of nearby symbols, and what is nearby in one language may be remote in another. Hofstadter considers the original poem and a French and a German translation, and shows how each translator adapts the linguistic form and even the contents of the poem in search for equivalent effects. This includes the invention of new words in the corresponding language, where the ingredient words employed by Carroll may not necessarily exist.

3.2 What Creativity is Involved

The composition of a whole poem involves creative decisions at various level of linguistic decision but the present analysis is concentrated on the level of innovation at the lexical level.

Suppose a baseline assignment of definitions to the elements of Wiggins' characterization:

- U the set of all orthographically valid words according to the rules of English
- R the set of rules that describe semantically meaningful words in English at the time the poem was written
- T the algorithm for lexical choice available to Lewis Carroll (memory, dictionary look up...)
- E the criteria used by readers to evaluate a particular word

This would have been the starting point at which Lewis Carroll found himself when he set out to write the poem. Hofstadter's analysis of the translation process may be related to the formalisation applied so far in the following way: each node in such a symbolic network would correspond to a concept, the process of learning a new concept corresponds to adding new nodes to the network and linking them appropriately by activation synapses to existing nodes. In terms of the formalism, our description now becomes:

- U the set of all possible nodes and the connections between them
- R the set of rules that describe nodes and connections associated with words of the language
- T the algorithm for travelling along the symbolic network in order to find a word

- E the criteria used by readers to evaluate the activation patterns resulting from the interpretation of a particular word

It is clear that a user may know a word (have it included in the set determined by R) and yet never use it himself when composing sentences (it is not accessible by means of his T). The introduction of a new word will require on one hand an extension of the existing set of nodes (R) but also an extension of the procedure for composing sentences (T) so that the word is actually used in production. What makes *Jabberwocky* a striking poem from the creative point of view is that Lewis Carroll has extended the conceptual space of words available for lexical choice beyond the set of words that have an accepted meaning in English. In terms of Wiggins' model, R has been extended to include newly formed words. however, the correct sequence to be considered is: Carroll extends his T set to include new techniques., the new techniques result in elements that are beyond R, (therefore requiring a new set R'). One option is to consider that transformational creativity has taken place at this stage, yet the sequence is not yet finished. The extended set R' (or the examples of it generated by Carroll's T) meet with approval from the literary taste of the time (score well under their E function). R' becomes the new R for this domain. The other option is to consider that transformational creativity has taken place only if this last effect is achieved. On additional consequence is that Carroll's T has by then become available to other authors, and yet intuitively it seems that any further application of the same technique would no longer be deemed transformationally creative. This could bring in to play the other dimension of Boden's classification: an act of creation could be either H-transformational or P-transformational (depending on whether the evaluators perception of R includes the result or not).

The evidence provided by Carroll himself, both in the words of his characters and in his later explanatory letters, provides an insight into the actual mechanisms that result from – or constitute a sketch of – the new set of rules T. It is apparent from the various attempts at explaining the techniques that the author does not have a clear picture of how exactly the new words have come about, but he is certainly aware of an assortment of possible mechanisms that are somehow involved in the process¹. Furthermore, Carroll has made his readers aware of these processes in his work. Two interesting questions arise from this last observation. What role does the fact that Carroll explains

¹ This seems to back up Wiggins' conjecture on the need for self awareness involved in creativity at the meta level.

his technique (in fact his new T) in his book play in the ensuing success (the high score for E)? What role does it play in making any further use of the same T a matter of exploratory rather than transformational creativity? The issue is not discussed further here, yet it has bearing on the interpretation of modern art and modern music, in which new creations seem to be hard to evaluate without an added explanation of how they came about.

The techniques described by Dean illustrate how the author has taken pains to ensure that the final product meets the constraints imposed by E. It is unclear to what extent the particular technique used for word creation assures that the readers will identify the meaning intended by the writer, but the success of the poem over time suggests that the author found a way of meeting the necessary requirements, even though the poem was built up by applying rules beyond those traditionally used by readers to evaluate poems. An important role is played by the interaction between the different levels of linguistic decision. Carroll makes sure that, while he is being wildly innovative at the lexical level, he remains conservative at the phonetic and syntactic levels.

Hofstadter argues that the networks corresponding to different languages are non-isomorphic, and that the challenge for the translator involves producing a similar activation pattern, even if it involves nodes that are not necessarily the formally correct equivalents of those activated by the original. This is true of translation in general. In cases as the particular one under discussion, there seems to be an additional ingredient: the invention of new words calls for the creation of new nodes and/or new activation synapses. Two basic issues to be considered in detail arise: how an author produces a new concept (if he ever actually does explicitly), and how a reader reacts on finding a new word (and whether he does produce a 'meaning' for it). Both processes seem to involve creative behaviour: having created a new word requires the creation of a meaning for it. For a case like the one described this can either be provided by the author in the accompanying text, or by the reader when trying to interpret the text. When no meaning is available, it is plausible to assume that the reader produces a tentative - and possibly partial - meaning that fits in with the interpretation of the rest of the text. There are certainly complex creative processes involved in the interpretation of unknown words in general, and this is one of the areas where having a reasonable formalisation of how creativity operates at the various levels may be most useful. One wonders whether there is any guarantee that, given both concept creations, the resulting concepts of author and reader bear resemblance at all, or simply produce a similar general impression. From Hofstadter's description of the translators

tricks of the trade, it seems that achieving such a 'similar general impression' is all that is required for a translation to be deemed correct.

4 Metaphor

The study of well documented instances of creativity at the semantic level may provide insights on this issue. One such instance is the use of metaphor in literary texts.

4.1 Theories of metaphor

Various studies (Gentner et al, 1989; Martin, 1990; Indurkha, 1992; Veale and Keane, 1993; Veale, 1995; Barnden, 1997) have been carried out in search for a computational theory of metaphor. A metaphor involves a conceptual transfer from one object of situation (the source or vehicle domain) and another object or situation (the target or tenor domain). Metaphorical interpretation is considered to be directional: each domain or object has a different role and its interchange will not lead to the same meaning (though it may yield an equally valuable metaphor). Metaphor is constrained by deep rules of coherency. Concepts that are mapped from one domain to another should be coherent among themselves (Indurkha, 1992). Some research on metaphor interpretation consists in finding the largest mapping function (between domains) that avoids inconsistencies.

An elementary structural description of metaphor interpretation is provided in the Sapper system (Veale, 1995). Starting from a representation of semantic memory in terms of a network of concepts linked together by associations, metaphor is described as a two-step process that first identifies implicit relationships between concepts (*dormant bridges*) and then establishes an explicit link between them (an *awakened bridge* that represents the metaphor in memory). In the Sapper system a rule-based symbolic solution deals with the first step and a spreading activation connectionist step deals with the second. Once a bridge is awakened, it effectively warps the memory so that the tenor and vehicle domains move conceptually together. The identification of dormant bridges is carried out not only in terms of identifying structure based on shared associations (Triangulation Rule) but also based on shared metaphor bridges (Squaring Rule).

The consistent transfer of concepts between two different domains is addressed in Leite et al (2000), where Dynamic Logic Programming is used to resolve the possible inconsistencies arising from the transfer, while retaining as much as possible of the added value represented by the metaphor. A metaphorical framework consists of two theories (tenor and vehicle), defined in two different languages, together with a function mapping one part of the language of the vehicle into the language of the tenor.

The final theory will consist of the tenor theory together those rules from the transformed vehicle theory that are not contradicted by the rules from the tenor theory.

4.2 Examples

The following lines from a poem provide an example of the type of metaphor employed in literature in general and poetry in particular:

the streetlamps were switched off
and the crickets were switched on

(Federico García Lorca, *Gipsy Ballads*)

Crickets are not switched on, but rather start singing at sunrise. The poet forces the use of the verb 'to switch on' together with the noun 'crickets', which is against the strict rules of language. The metaphor works (can be easily understood by the reader) because crickets are

Crickets	Streetlights
start and stop at periodic intervals	start and stop at periodic intervals
feature in our perception of a scene	feature in our perception of a scene
	switch on and off
animate	inanimate
active	passive
natural	man-made

Table 1. 'Theories' for crickets and streetlights

known to start and stop singing at regular intervals, just as if they were switched on and off by a mysterious hand. The transgression of the rules of language is licensed by the effect achieved: a parallel is drawn in this way between the two verses, in the form of a pattern that is followed by the sentences that make up each one.

One possible interpretation of this example in terms of theory extension (Leite et al, 2000) would be to consider the domain of crickets as tenor and the domain of streetlights as vehicle. The concepts involved for each domain might be represented as shown in table 1.

The fact that both theories share concepts such as starting and stopping at periodic intervals, or being features in our perception of a scene, provides the crossover point, which allows the concept of being switched on and off to be applied to crickets. This would require all other inconsistent concepts (animate/inanimate, active/passive, natural/man-made) associated with switching on and off a streetlight to be pruned from the meaning it has in the full sentence. However, once the rules are broken, it is not clear whether the poet intends the reader to imagine

that the crickets are a man-made addition to the landscape, or that the switching off of the streetlamps (or their very existence) is as natural as the song of the crickets. This observation brings into question the directionality of metaphor in a strictly linguistic literary use. In more general terms, because the rules have been broken in this way, each reader may end up with a different reconstruction of the poet's intention. It may be this multiplicity of possible interpretations that makes a metaphor specially successful.

A different example from the same source shows similar behaviour:

Playing her parchment moon
Preciosa comes along...

Preciosa is a young gypsy girl. She has no moon, and there is no moon made of parchment, but the moon is round, and she is playing a tambourine (which is round like the moon and made of parchent). The 'theory' for the domain of tambourines (tenor) and the domain of moons (vehicle) might be represented as shown in table 2.

Tambourine	Moon
round	round
parchment	
played on	
active	passive
belongs to Preciosa	
	admired by all

Table 2. 'Theories' for tambourines and moons

It is interesting to note that here the part of the concept that actually establishes the bridge between the domains in both examples is actually not present in the text. Again, the description of what is actually used as a bridge may in fact be different for different readers, or may simply be a 'similar general impression'.

Finally, the actual interplay between concepts that is the hallmark of a poet can be more complex than the sort of simple crossover described in the examples so far. Take the following example by the same author:

Against the bitter green,
a card-hard light
traces raging horses
and riders' silhouettes.

This occurs in a context where a knife fight takes place in an olive grove between men on horseback for an unknown reason, and ends in the death of several of them.

Various interpretation can be put upon the various concepts interacting in these verses by literary critics. Harris (1991) believes that the reference to playing cards in the second line indicates "a reason for the fight", and shows "the quality of hardness from the knives" transferred to the light. Havard (1990) points out that "cards depict motifs in profile, and the equivalent of a jack in Spanish cards is a horse and rider". Whether one accepts this depth of analysis as meaningful or not, it is clear that there are hidden interactions between the concepts represented by the words of the poem well beyond those that could be sketched along the lines followed above. The bitterness of the situation is attributed to the colour, a hypothetical reason for the fight (neither playing cards nor gambling feature anywhere else in the poem) is used to qualify the hardness (supposedly of the knives) attributed to the light, and the scene described evokes the picture that appears in a playing card which has been brought in to qualify the light and act as suggested reason for the fight. How many of those were intended by the author, how many arose by chance (but were retained by the sensibility of the poet as good contributions), how many did not even surface into the author's consciousness... these are questions for which no answers are available, and yet, from the moment that someone points out the interactions, there is a certain need to model them and provide a computational theory to explain/achieve/evaluate them.

4.4 Which Creativity

One possible baseline assignment of definitions to the elements of Wiggins' characterization would be:

- U the set of all syntactically valid word combinations according to the rules of Spanish
- R the set of rules that describe semantically meaningful combinations of words in Spanish at the time the poem was written
- T the algorithm for sentence composition available to Lorca
- E the criteria used by readers to evaluate the resulting sentence

It is not particularly clear what should be understood as 'semantically meaningful combinations of words in Spanish'. Certainly metaphor had been used often before in literature, so metaphorical uses of particular words and a number of typical word combinations for constructing metaphors should be considered as already included in the set defined by R, and the mechanisms for using metaphor as included in T. In this case, it appears that any particular innovation introduced by Lorca would have to be interpreted solely in terms of exploratory creativity, his creative contribution taking the form of making explicit particular combinations of words that were possibly not

available by application of the individual composition techniques of previous authors.

Why is metaphor so striking in general if it in essence it reduces to simple word combination? The assignment given may have been drawn too broadly. The example can be analysed at a different level of granularity, by focusing the description in terms of Wiggins' septuple to a different level of linguistic decision. Consider the following alternative assignment of definitions:

- U the set of all possible definitions of the meanings for the words and sentences of Spanish (including partial definitions)
- R the set of rules that describe the particular meanings assigned to Spanish words and sentences at the time the poem was written
- T the compositional algorithm for constructing the meaning of a sentence from the meanings of the words used to construct it (as employed by Lorca)
- E the criteria used by readers to evaluate the resulting sentence

In this case, it may be possible that the T applied by Lorca results in a different set of (surviving partial) meanings within the context of this sentences for the words he actually employs. this interpretation fits better with our perception of poetry and its effect on us. However, it has severe implications on the feasibility of formalising meaning, and on long standing assumptions such as the principle of compositionality of language.

5. Conclusions

Various levels of linguistic decision have been shown at play in two examples of creativity. The first example showed how creative behaviour does not occur in the same degree across all levels. Rather, a conservative approach in some levels is required for a successful interpretation of creative innovations at other levels. Creativity at a linguistic level can be counterproductive for communication if abused. This suggests that different levels of linguistic decision should not be exploited creatively at the same time. With respect to the question asked at the beginning of the paper, the creativity of a sentence may be evaluated at the different levels of linguistic decision. For each set of examples, the analysis of the creative behaviour in terms of exploration of a conceptual space has been used. In order to capture the peculiarities of creative behaviour intuitively associated with the examples, the conceptual spaces involved must be restricted to the domains of application presented by the level of linguistic decision being creatively exploited.

To formalise this idea, a different conceptual space would have to be used for each different level of linguistic decision. The overall representation for the conceptual space of language use would involve the union of the different concepts. This fits in with the idea that there is no homogeneous theory of language, but rather a set of theories, each one governing a different level of linguistic decision.

An additional problem that would have to be tackled is the extent to which the interaction between the theories for the different levels complicates the picture significantly. Intuition suggests that it will to a considerable extent. Creativity may operate at each of the levels of decision involved in linguistic production, but it may interact between different levels in ways that are not evident. As it has been shown in the ingenious use of phonetic restrictions to support lexical invention in Carroll's poem, an additional level of creativity comes about when some form of interplay between different levels of creativity appears - whether consciously or unconsciously, as explained by the author's misgivings as to the validity of his own explanations of his effects (Graham, 1981).

It is important to consider that, because of the nested structure of the levels of decision described, creativity at a certain level may imply creativity at the level below it. For instance, inventing a new word may be done according to the accepted rules of word formation (even if the valid particular combination of valid syllables had not been used in the language before) or, because it is new, it may be done while breaking accepted rules of syllable construction. This last example would involve creativity at the lexical and the phonetic level. At a finer level of analysis, the creation of words in *Jabberwocky* involves exploratory creativity at the phonetic level and transformational creativity at the lexical level. The creation of words involving new rules of syllable construction would count as transformational creativity on both levels. There is a possibility that it also involve transformational creativity at the semantic level, if a meaning that could not be represented in the language before is now represented by this word. There may be transformational creativity at the syntactic level if a new syntactic category is invented. Another aspect that is at play in determining whether a sentence is creative is the effect of the point of view of the evaluator. Three issues must be considered: whether the speaker or writer considers he has been creative in producing that sentence (Boden's P-creativity), whether other people consider the sentence creative (related with Boden's H-creativity and Wiggins's E function), and whether the sentence can be considered a valid sentence of the language (related to Wiggins' R rules). This question acquires particular importance in the study of linguistic creativity, since there are neither universally agreed rules

for establishing the validity of a language expression, nor recognised ways of establishing consensus between speakers of a language that are consistently used when accepting a new word or meaning into the language. It is still an open issue whether a conceptual space is deemed to have been transformed if somebody uses a new word or whether this can be taken to have happened only if somebody uses it successfully (i.e. manages to convey the intended meaning to somebody else). The extreme criterion would be to consider such creativity to have taken place only if the new word becomes generally accepted and used in the language.

References

- Ball, H., *Flight Out of Time*. New York, Viking Press, 1974.
- Barnden, J.A., An AI system for metaphorical reasoning about mental states in discourse. In J-P. Koenig (Ed.) *Conceptual Structure, Discourse and Language II*, Stanford, Ca., 1997.
- Boden, M., *The Creative Mind*, Abacus, 1990.
- Boden, M., Creativity and Artificial Intelligence, *Artificial Intelligence*, 103: 347-356, 1998.
- Bringsjord, S. and Ferruci, D., 2000. *Artificial Intelligence and Literary Creativity*, Lawrence Erlbaum Associates, New York.
- Carroll, L., 1872. *Through the Looking-Glass and What Alice Found There*, 1872
- Dean, C., 2001, *The Jabberwocky*, <http://home.earthlink.net/lfdean/carroll/>
- Espy, W.R., *The Wordsworth Rhyming dictionary*, Wordsworth Editions, Hertfordshire, 1997.
- García Lorca, F., *Romancero gitano*.
- Gentner, D., Falkenheimer, B. and Skorstad, J., *Metaphor: The Good, The Bad and The Ugly*. In Y. Wilks (Ed.) *Theoretical Issues in Natural Language Processing*. Hillsdale, NJ: Lawrence Erlbaum Associates, 1989.
- Graham, E. "Lewis Carroll and the Writing of *Through the Looking Glass*", Introduction to *Through the Looking Glass*. In *Alice's Adventures in Wonderland/Through the Looking Glass*, Puffin Books, Great Britain, 1981.
- Hausmann, R., 'The Ortophonic Dawn', Stereo Headphones No. 4, Spring, Suffolk, England: Nicholas Zurbrugg, 1971.
- Hofstadter, D.R., Gödel, Escher, Bach: *An Eternal Golden Braid*, New York: Basic Books, 1980; Vintage Books Edition, Sep 1980.

- Hultberg, T., (Ed.), *Literally Speaking: sound poetry & text-sound composition*, Sweden: Bo Ejeby Edition, 1993.
- Indurkha, B., *Metaphor and Cognition*, Kluwer Academic Publishers, Dordrecht, 1992.
- Martin, J.H., *A computational model of metaphor interpretation*, Academic Press, 1990
- Meehan, J. *The metanovel: writing stories by computer*. Ann Arbor: University Microfilms International, 1977
- Leite, J.A., Pereira, F.C., Cardoso, A. and Pereira, L.M., *Metaphorical mapping consistency via Dynamic Logic Programming*. In *Time for AI and Society, Proceeding of the AISB'00 Symposium on Creative & Cultural Aspects and Applications of AI & Cognitive Science*, 17-20th April, 2000, University of Birmingham.
- Poe, E.A., *La filosofía de la composición, seguida de El cuervo*. Ediciones Coyoacán, México, 1997.
- Ritchie, G., *Describing verbally expressed humour*. In *Time for AI and Society, Proceeding of the AISB'00 Symposium on Creative & Cultural Aspects and Applications of AI & Cognitive Science*, 17-20th April, 2000, University of Birmingham.
- Ritchie, G., *Assessing creativity*. In: *Proceedings of the AISB'01 Symposium on Artificial Intelligence and Creativity in Arts and Science*, 21st-24th March 2001, University of York.
- Schwitters, K., 'From MERZ 1920'. In: J. Rothenberg and P. Joris (Eds.) *pppppp: POEMS PERFORMANCES PIECES PROSES PLAYS POETICS*, Philadelphia: Temple University Press, 1993.
- Turner, S. *MINSTREL: a computer model of creativity and storytelling*. PhD Thesis, Computer Science Department, University of California, Los Angeles. Technical Report CSD-920057, 1992.
- Veale, T., *Metaphor, Memory and Meaning: Symbolic and Connectionist Issues in Metaphor Comprehension*. PhD Thesis, Trinity College, Dublin, 1995.
- Veale, T. and M. Keane, *A Connectionist Model of Semantic Memory for Metaphor Interpretation*, 1993 Workshop on Neural Architectures and Distributed AI.
- Wiggins, G.A., *Towards a More Precise Characterization of Creativity in AI*. In: R. Weber and C. Gresse von Wangenheim (Eds.), *Proceedings of the Workshop Program at the Fourth International Conference on Case-Based Reasoning 2001*, Technical Note AIC-01-003, Washington, DC: Naval Research Laboratory, Navy Center for Applied Research in Artificial Intelligence.

AWAITING THE SENSATION OF A SHORT, SHARP SHOCK: TWIST-CENTRED STORY GENERATION BY TRANSFORMATION

John Platts¹, Ann Blandford² & Christian Huyck¹

1: School of Computing Science, Middlesex University, Bounds Green Road, London N11 2NQ

2: UCL Interaction Centre, UCL, 26, Bedford Way, London, WC1H 0AB

{j.platts;c.huyck}@mdx.ac.uk, a.blandford@ucl.ac.uk

Abstract

A mechanism whereby a human storyteller may compose a twist-centred story, that is, a story possessing a surprising ending that obliges the addressee (audience or reader) to reappraise their understanding of what has preceded it, is proposed. The mechanism involves the generation of an overt story, its conversion into a concealed story, and the building of a twist phase whose function is to destroy the credibility of the overt story and reveal the concealed story. A piece of software, TWISTER, which models this process, is described. Examples of its operation are given.

1 Introduction

This paper is concerned with the generation of twist-centred stories - stories, whether spoken, written or performed, which have a surprise, "twist", ending. It seems clear that successful authors and storytellers frequently produce such compositions, and that the ability to do so is particularly valued in such authors.

In this paper, it is contended that a successful twist-centred story can be generated by the following process. An existing story is transformed into two variants, known as "the overt story" and "the concealed story". These two components are synthesised into an overall story, presented in such a way that the addressee (audience or reader) interprets what they are reading as the overt story. A "twist phase" is then introduced, which has the effect of making the audience realise that what they are reading is in fact the concealed story. It may also be necessary to introduce a "post-twist phase", in which issues raised by the twist phase are satisfactorily resolved. In this paper, the characteristics of these components, and the processes that generate them, are examined. Computational examples of parts of these processes are provided. Finally, the degree to which this is a satisfactory model of the way in which human storytellers generate twist-centred stories is discussed.

2 The Seeds of Stories

If one wishes to model the process whereby a human storyteller generates an effective story, one is faced with the question of what the starting point for such a story - the seed from which the story is to be grown -

should be. Earlier authors in the field of automated story generation have tended to follow one of two strategies. Meehan (1976), and those who have developed his work, used a scheme in which the storyteller invented a setting, and characters, and a problem for the central character, and the story took the form of a report of the character's efforts to extricate himself from the problem. Turner (1994), in contrast, used a scheme in which some well-known piece of conventional wisdom, such as "pride comes before a fall", was instantiated with characters and events in order to make it into a story, and further events or background items were added to make the story coherent and believable. Bringsjord & Ferrucci (2000), in a variation on this approach, started with a logical definition of one of the compelling themes to be found in literature, namely "betrayal", rather than a piece of conventional wisdom, and instantiated it in a similar way.

In this paper, we suggest that an alternative approach, which a storyteller may very well find more amenable, is to start with an existing story, extract its essence, and transform it into a surprising (and therefore original) form. In support of this hypothesis, we have given expert storytellers an example of a story fragment taken from Irish mythology - "the raven banner" - and asked them how they would turn it into a story.

The material presented was as follows:

The Raven Banner was a magic flag that a King could have carried before him into battle. If he did so, he was bound to win the battle. But the warrior who carried the banner was bound to be killed in the battle.

On occasions, they generated stories with twists. In other words, the author started with a story

fragment, an overt story. They then tried to find a way to manipulate the story to achieve a surprising end. All of the basic constraints of the overt story would be met: there is a magic banner which is carried into battle; the warrior who carries it is doomed to die, and does so; and the side he is on wins, as prophesied. However, the storyteller would ingeniously introduce extra features of the story, not revealed until the end, that confounded the expectations that the reader possessed (or rather, might reasonably be expected to possess). For instance, one author suggested that, after the warrior had died, and victory had been assured, the king discovered that the warrior was, in fact, his (the king's) sole heir. A concealed story is generated (involving the warrior's true identity, and why it is unknown to the king), and so is a twist phase (the king's, and therefore the reader's, discovery). The function of the twist phase is to reveal the concealed story to the reader, in such a way that it replaces the overt story (in which a relatively unimportant warrior makes the sacrifice) in their comprehension.

3 The Mechanism

We are currently developing a piece of software, TWISTER, whose function will be to generate twist-centred stories of the type described above. The process involves taking an existing story, and dividing it into episodes, then explaining what is happening in these episodes. This provides an overt story. Then an alternative explanation is generated, and, from this, alternative episodes, thus generating a concealed story. Conflicts between the two versions of the story are detected and eliminated. A twist phase is generated, based on information in the concealed story. It may be necessary to generate a post-twist phase. Finally, some of these components – the overt story as far as the climax, the twist phase, the post-twist phase – are assembled into an overall story. The effect of the overall story so generated should be to mislead the reader into accepting the overt version of the story until the twist phase is reached. The system has, however, generated an alternate concealed version of the story that is also plausible. During the twist phase, the reader is obliged to abandon their comprehension of the overt story, and construct a comprehension of the concealed story, on the basis of what they now know. What the reader experiences will, with any luck, be surprise.

This process is explored in more detail in section 3.1. The way in which the process is implemented in TWISTER is described in section 3.2. Finally, section 3.3 describes a slightly different approach to the provision of the seed story.

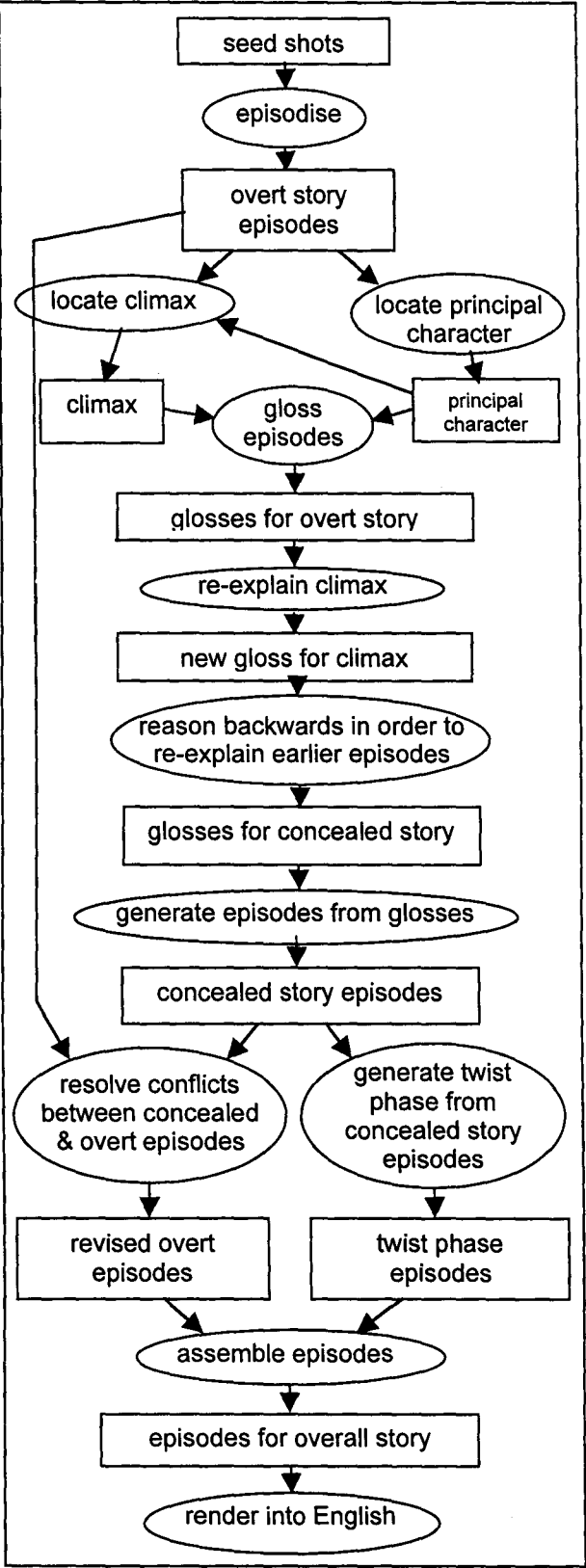


Fig.1: The processes involved in the simple version of TWISTER. Following the diagrammatic conventions of KADS (Tansley & Hayball, 1993: p.74), inference steps are shown in elliptical boxes, and the products of such inference steps are shown in rectangular boxes.

3.1 Simple story-telling by transformation, and how TWISTER embodies it

At present, TWISTER takes as its input a semantic representation of a simple story: a collection of "story shots". Each of these "shots" represents a simple action or a simple description. The process of converting this into a twist-centred story is described in Fig.1. Parts of this process have been implemented, other parts are still under development.

TWISTER's first task is to divide the story into episodes. Next, it must explain (we use the term "gloss") the various episodes in the story; these glosses are stored internally rather than being presented to the reader. Then the episodes are re-explained: alternative glosses are generated for the episodes, starting with the one identified as the climax. This is a process of backward reasoning, in that the climax may be seen as the culmination of reasoning processes in the story, and re-explaining follows these processes backwards towards their initial conditions. As it does so, it generates episodes necessary for these new glosses to make sense. These new episodes form the "concealed" story. Any contradictions between the concealed episodes and the overt episodes are resolved, by amending the overt episode. A twist phase is generated, in the form of a plausible reason to reveal one or more of the concealed episodes. The story can then be related to the reader, in the form of the overt story episodes, (some of which may have been amended), followed by the twist phase episodes.

3.2 Computational examples

TWISTER is written in Prolog, and consists of a number of modules. It contains a lexicon of 654 verbs, 41 nouns, and 11 adjectives. An ontology of verb types has been written, containing 195 nodes, and the verb meanings are organised on the basis of this. A small portion of this ontology is shown in fig. 2. It enables

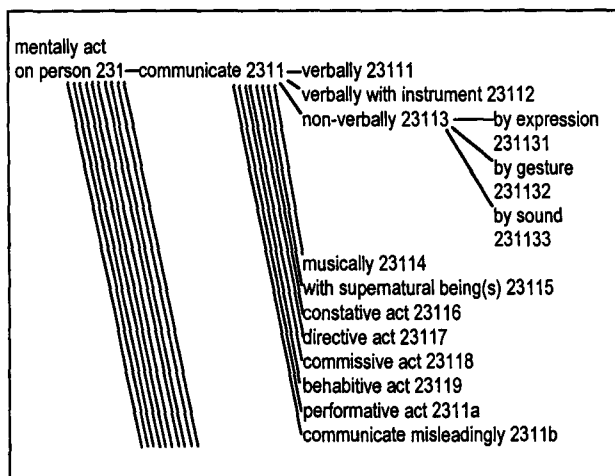


Fig.2: Part of the verb ontology used in TWISTER.

TWISTER to reason on the basis of hierarchies of actions and descriptions. The lexicon also contains data (mainly concerned with sentence frames) derived from WordNet, the electronic lexical database written at Princeton University (see Fellbaum (1998) for a description of WordNet).

```

statement(overt, 1, rel, semrep('rain',
[[1,0,_, p, 1000, 1020, [neutral], p],
[[time,'on','Wednesday','then',_,_]]])).
statement(overt, 2, rel, semrep('walk',
[[1,0,_, i, 1010, 1010, [neutral], p],
[[ag,'by','Pat','he',3,s]],
[goal,'into','the bank','there',_,_]]])).
statement(overt, 3, rel, semrep('approach',
[[1,0,_, i, 1011, 1011, [neutral], p],
[[ag,'by','Pat','he',3,s]], [goal,'the
counter','there',_,_]]])).
statement(overt, 4, rel, semrep('ask',
[[3,0,1], i, 1012, 1012, [neutral], p],
[[ag,'by','Pat','he',3,s]], [exp,'the
teller','she',3,s]], [utt,_
statement(overt, 5), _]]])).
statement(overt, 5, nonrel, semrep('cash',
[[1,0,_, i, 1013, 1013, [modal(future, _)],
p], [[ag,'by','the teller','she',3,s]],
[exp,'for','Pat','he',3,s]], [thm,
'', 'a cheque', ['it',3,s]]])).
statement(overt, 6, rel, semrep('say',
[[3,0,1], i, 1013, 1013, [neutral], p],
[[ag,'by','the teller','she',3,s]],
[exp,'to','Pat','he',3,s]], [utt,_
statement(overt, 7), _]]])).
statement(overt, 7, nonrel, semrep('own',
[[1,0,_, i, 1013, 1013, [neutral], n],
[[ag,'by','Pat','he',3,s]], [thm, '', 'any
money', ['it',3,s]]])).
statement(overt, 8, rel, semrep('become',
[[1,0,_, i, 1014, 1014, [neutral], p],
[[ag,'by','Pat','he',3,s]], [thm, '',
'angry', ['it',3,s]]])).
statement(overt, 9, rel, semrep('shout',
[[3,0,7], i, 1015, 1015, [neutral], p],
[[ag,'by','Pat','he',3,s]]])).
statement(overt, 10, rel, semrep('cry',
[[1,0,_, i, 1016, 1016, [neutral], p],
[[ag,'by','Pat','he',3,s]]])).

```

Fig. 3: The shots making up the story "Pat goes to the bank", expressed in TWISTER's semantic representation

Fig. 3 shows a simple ten shot story, in the semantic representation format that TWISTER uses. If given this collection of statements, "Render", the language-generation module in TWISTER, would translate it as follows:

It was raining on Wednesday.
Pat walked into the bank.
Pat approached the counter.
Pat asked the teller to cash a cheque for Pat.
The teller said to Pat that Pat didn't own any money.
Pat became angry.
Pat shouted.
Pat cried.

When asked to episodise this story, TWISTER uses some rather simple principles (involving adjacency, similarity of actions, and involvement of a particular character) to produce the structures shown in fig.4.

```

Episodes corresponding to the overt story -
episode: overt, 1 starting at time 1000,
finishing at time 1020
  the type is: [scSet]
  consisting of shots -
    It was raining on Wednesday.
episode: overt, 2 starting at time 1010,
finishing at time 1011
  the type is: [move]
  consisting of shots -
    Pat walked into the bank.
    Pat approached the counter.
episode: overt, 3 starting at time 1012,
finishing at time 1013
  the type is: [talk]
  consisting of shots -
    Pat asked the teller to cash a cheque
    for Pat.
    The teller said to Pat that Pat
    didn't own any money.
episode: overt, 4 starting at time 1015,
finishing at time 1016
  the type is: [drama]
  consisting of shots -
    Pat shouted.
    Pat cried.

```

Fig.4: episodes in the story "Pat goes to the bank", as identified by TWISTER.

TWISTER also identifies Pat as the principal character (because of the frequency with which he is involved in actions) and the last episode as the climax (because of its position in the sequence, coupled with the fact that something "dramatic" is happening to the principal character in it).

Subsequent reasoning stages are based on the use of "lexical scripts": stock descriptions of actions, containing both pre- and post-conditions. These have much in common with Schank & Abelson's "scripts" (Schank & Abelson, 1977) and Correira's "extended Horn clause rules" (Correira, 1980). They allow a story to be extended backwards, from a particular episode or gloss, by a process that takes the episode/gloss and a set of lexical scripts and selects the scripts that lead to this set of statements.

When asked to produce a gloss for the second of the episodes listed in fig.4, TWISTER provides the following:

```

gloss attached to episode no. overt, 2
contains the following elements -
  Pat had a plan at 1010.
  Pat had to be at the counter at 1010.

```

Fig.5: A sample episode gloss.

However, after the next episode has been glossed, this episode 2 gloss is augmented:

```

gloss attached to episode no. overt, 2
contains the following elements -
  Pat had a plan at 1010.
  Pat had to be at the counter at 1010.
  The plan was that Pat would take
  money from Pat's bank account.

```

Fig.6: The same gloss, with an extra feature.

The reasoning here is more complex than before. First, (using a low-level reasoning mechanism, not based on lexical scripts), TWISTER recognises that Pat has gone into a location, and is still there, and that it is a place where "finance" takes place. Second, a lexical script is located that (in effect) says that, if a person goes into such a place, they have a plan that involves them being there. Third, and on the basis of information extracted from the next episode, a lexical script is located that (in effect) says if one directs someone to do something, this is in fulfilment of one's plan. Fourth, another lexical script is located that (in effect) says that to cash a cheque has the result that one has taken money from one's bank account. And last, another lexical script is located that (in effect) says that if one is in a place where "finance" takes place, and one directs a person who is "financially authorised" to cash a cheque, and one has money in the account, then one *can* successfully cash a cheque.

It should be clear that, though the reasoning may be long and involved, lexical-script-based reasoning (augmented with lower-level reasoning) can explain episodes (provided that there are sufficient lexical scripts in the library). Re-explaining is essentially similar, in that an episode must be given a gloss, but one or more of the lexscripts used in the previous glossing stage are explicitly disallowed. However, re-explaining is coupled to a backward reasoning process. This means that evidence is sought in the episode that is being re-explained, and prior to this episode, to fulfil the pre-conditions of the lexical script. If the available facts (in the episodes and glosses) contradict these pre-conditions, then the lexical script must be rejected. But if they are merely absent (rather than present in a contradictory form), then the facts can be invented, with details chosen from files of suitable instantiations, and recorded as glosses that belong to the concealed story. Given the glosses, accompanying episodes (that also belong to the concealed story) can be invented.

It is intended that, given a gloss to the climax of this story that specifies that Pat is upset, because Pat thinks that someone has stolen the money that was in his bank account, TWISTER should come up with a complete concealed story. This would involve Pat *pretending* to be upset. The "Pat is upset" statement is slightly changed; the attached episode - overt 4 - doesn't have to be. The "because" statement is replaced completely (the lexical script that generated it is no longer viable): a lexical script is located that says (in effect) that one pretends to be upset because one wishes to cause a

diversion, i.e. the act of pretending to be upset can have the effect of making persons not notice that an act (undesirable from their point of view) is taking place. The existence of this act, at the same time as the diversion, is a pre-condition. Backward reasoning must now invent this act, undesirable to the teller. Lexical-script-based reasoning could devise "a robbery", though low-level reasoning would be needed to establish that it wasn't Pat who was doing it, and to invent a new character ("Sandy", perhaps) who *was* doing it. Backward reasoning would proceed to invent episodes in which Sandy went in to the bank and stole the money – there are no facts in the overt story to contradict the idea that he went into the bank, and the reasoning that would conclude that someone would stop him from stealing the money is de-activated by the statement about the diversion.

Backward reasoning from the "causing a diversion" script would also eventually lead to the conclusion that Pat must arrange for there to be no money in his bank account. Another lexical script can be used to generate a concealed episode in which Pat removed the money at an earlier time.

A search for conflicts between the newly-created concealed story and the overt story would reveal that Pat has a different plan in the two versions, but this is not a conflict that needs to be resolved – in general, a character can have differing cognitions of any sort in the two versions of the story, and it will not be treated as a problem.

There remains the problem of a twist phase. One strategy for generating this is to develop the concealed story beyond the climax that was the starting point for its generation. The concealed story glosses contain the information that Pat and Sandy had a plan to (collaboratively and surreptitiously) rob the bank. A re-examination of the lexical script that generated this gloss reveals the later phases of such a plan – the culprits leave the scene of the crime, and the culprits share the proceeds amongst themselves. A process of forward reasoning can be used to invent episodes that fulfil this script, beyond those that already exist in the concealed story. These are episodes in which Pat and Sandy both leave the bank, Pat and Sandy meet up, and Pat and Sandy divide the stolen money between them. Some of these can be revealed, to provide a simple twist phase – the simplest strategy is to reveal all of them. The final composition might be something like the following:

It was raining on Wednesday.
Pat walked into the bank.
Pat approached the counter.
Pat asked the teller to cash a cheque for Pat.
The teller said to Pat that Pat didn't own any money.
Pat became angry.

Pat shouted.
Pat cried.
Sandy left the bank.
Pat left the bank.
Pat and Sandy met at the street corner.
Pat and Sandy divided the money they had stolen.

This would be a better story with a statement such as "No one noticed what Sandy was doing" after the 8th line but, at the moment, it's not clear how such a statement could be generated.

The later stages of this description have been speculative. Nevertheless, it is contended that the mechanisms needed for re-examining, backward reasoning, and forward reasoning are simply variants on the mechanisms that already exist for glossing.

3.3 An alternative starting point

As a model of human storytelling, the approach described in section 3.1 assumes that the storyteller starts with a complete story in their minds, and then modifies it. It seems more plausible to assume that the storyteller will sometimes start from a more abstract version of the seed story. An author, wishing, perhaps, to "retell the Romeo and Juliet story in a new setting" will probably not start with a detailed version of Romeo and Juliet. Rather, they will start with an outline, containing the most important features. We have therefore given some consideration to the generation of a twist-centred story from a template.

A template is a set of seed episodes *generalised* from an existing story. For our example, we will take a story template from the Greek myth of Hesione and Heracles (Graves, 1955, p.169). A brief summary of the story is that Laomedon must sacrifice Hesione, his daughter, to a sea monster sent by Neptune, by chaining her to a rock on the sea-shore, ready for the monster to eat. Heracles arrives and releases her, making her grateful to him. Generalising away from individuals, and specific props, and generalising towards vaguer versions of the actions, the template is that:

- B causes C to be in danger from D
- As a result, C is in a dangerous situation.
- E gets C out of the dangerous situation.
- C is grateful to E.

It is also possible to make changes to the template, by eliminating dispensable characters (amongst other possibilities). The motivation might be to produce a story that is *not* simply a re-telling of the old story. If we do this to the template above, we have:

- C is in danger from D
- As a result, C is in a dangerous situation.
- E gets C out of the dangerous situation.
- C is grateful to E.

This story is then instantiated into a simple overt story. This involves providing C, D and E with names and characters, and rationalising why C is in the dangerous situation, why D is a threat, and why C, D and E are all there at the same time. It also extends the template forward to relate what happens next. For example, in the process of this rationalisation, we might generate the facts that C is weak, D is insane, E is helpful, D is unattractive, and the place where they all are is beautiful countryside. We can also give the characters names, such as Chloe, Darren, and Edward, and make Wensleydale the location. This of course is just one possible set of instantiations. With this instantiation, one possible alternative story goes as follows. Darren is insane (and dangerous) and walking around in Wensleydale. Chloe goes for a walk in Wensleydale and is cornered and threatened by Darren. Edward comes along at just the right moment to interrupt and drive Darren away.

All these instantiations can be performed by low-level reasoning, consulting files of ready-made details (such as a file of proper names, classified by gender), and, above all, matching against a library of lexical scripts as mentioned in the last section. These lexical scripts, coupled with a backward-reasoning mechanism, provide the essential component of the search. Each has its own preconditions, so new story facts may be derived, and existing story facts can be incorporated into the developing fabric of the story.

For example, in the story of the encounter in Wensleydale, we might infer that Chloe enjoys walking and that it is a beautiful day for a walk, and that is sufficient to account for her being in Wensleydale that day. A similar line of reasoning might be applied to account for why Edward is there. For Darren, an adequate account might be based on him having recently escaped from a mental hospital and now being on the run.

The system may have the knowledge, in the form of a lexical script, that someone is in a dangerous situation if some other person plans to physically attack them. The system will find the climax, *Chloe is in a dangerous situation* and try to derive a reason for it. Using the lexical scripts along with the existing story facts, the system extends the story back from provided knowledge like *Chloe is in a dangerous situation* to invented knowledge like *Darren plans to physically attack Chloe because he is insane and therefore Chloe is in a dangerous situation*.

The advantage of deriving invented knowledge from the climax as a principal story generating mechanism is that it guarantees that the story will reach the desired climax. The search space is diminished, and consistency and plausibility are ensured. As demonstrated in the previous section, deriving the

implicit knowledge using lexical scripts and the climax permits the generation of a complete story from a group of seed episodes. However, it should be appreciated that it is necessary to include in the system principles that determine when events in the story have been *sufficiently* explained, and *sufficiently* extended, otherwise the story will not have a satisfactory start or finish.

The overt story, so generated, will be plausible but is likely to be uninteresting. Both qualities arise from the use of lexical scripts: the reader knows the script, as it is relatively common knowledge, and finds an instantiation of it unexceptional. The next phase transforms the overt story into a concealed story. As explained in earlier sections, the concealed story is an alternative series of events that lead to the climatic episode; however the concealed story differs from the overt story in special ways that ensure that it is perceived as both substantially different to and (ideally) more interesting than the overt story.

It is contended that storytellers use a repertoire of tricks in order to generate such variant stories. One such trick is to rotate the roles played by the various characters and invent some reason why these roles were not immediately apparent. In the example given above, the original roles could be identified as victim (C), persecutor (D) and rescuer (E), and these might be manipulated by swapping persecutor and rescuer. Then the system needs to generate a reason why the overt facts do not support this interpretation. It can do so by using a lexical script involving *misapprehension* on Chloe's part (in contrast to the script involving *deception* on Pat's part, in the "Pat goes to the bank" story). The effect would be that Chloe misinterpreted Darren and Edward's behaviour, thinking that Darren was the persecutor and Edward the rescuer, when the facts were the reverse.

This is a particular case of an approach in which actions *appear* to be the same as those to be found in the overt story, when interpreted by a character within the story, but are in fact different in terms of unobservable features, notably motivation and intention. Thus, in the example, Chloe *believes* Darren to be dangerous and Edward to be the rescuer. The twist in the story is achieved by exposing the fact that these beliefs are incorrect: that Darren is – or could be – the rescuer, while Edward is dangerous.

A more general approach is to choose lexical scripts of lower plausibility than those used to construct the overt story from the seed episodes. Without an obvious mechanism for calculating plausibility in the context of a particular story, this seems an unpromising direction, but it might be possible to include a rating of default plausibility in the lexical script record.

When generating the concealed story, it is necessary to maintain a correspondence between it and the overt story, amending the episodes in one or both, so that when either is related, the events are not inconsistent with the other. It is also necessary for the chains of preconditions in both stories to be maintained - each story must remain self-consistent.

After the concealed story has been generated, we are left with two sets of plausible reasons for the story facts. The overt story is more plausible and should be the one that the reader has in mind at the climax. It is also necessary to generate a twist phase to the story.

The twist phase consists of one or more episodes, designed to replace the climactic episodes of the overt story, in which the props supporting the overt story are kicked away, and the concealed story is revealed. That is to say, certain preconditions without which the overt story is unsupportable are revealed to be untrue, and certain pieces of evidence that point particularly strongly to one or more of the episodes to be found exclusively in the concealed story are presented.

It may also be necessary to generate a post-twist phase. In this, forward-chaining is used to generate the episodes that immediately follow the twist phase. Similarly existing facts may also be used as preconditions for lexical scripts to derive future story events.

Edward can be revealed as being insane by having him say something that exposes that fact. For example "I am the Duke of Wensleydale".

4 Discussion

It was suggested earlier that many human storytellers are accustomed to generating twist-centred stories. The mechanism described above amounts to a model of such story-generating behaviour. The degree to which it can be considered a good model has yet to be established. Further work with practicing authors might throw light upon this question. It would be desirable to use the techniques of knowledge elicitation to discover:

- a) whether such practitioners, when endeavouring to compose a twist-centred story, do in fact locate the climax of what they have already written and reason backwards from it;
 - b) whether they do in fact create an overt story, and then a parallel concealed story (and, if so, whether they invariably do so);
 - c) whether they are using something approximating to a lexical script in their reasoning and, if so, how they are tackling questions of plausibility;
- among other questions.

Important issues raised by this study include the nature of a substantial (as opposed to trivial) transformation of an episode, the nature of a surprising episode, the nature of plausibility in the context of storytelling, and how these qualities can be manipulated by a program.

It may be possible to draw up a list of features that must be present if the transformation of an episode is to be considered substantial. Or it might be possible to arrange that, if an episode is to be transformed into another, substantially different, episode, the process involves the utilisation of a lexical script that is distinctly different from the script used to gloss the first episode. This would be possible if the lexical scripts were organised hierarchically, so that it was possible to measure the semantic distance between them.

Plausibility (or *vraisemblance*, to use a term more widely found in the field of literary studies) can perhaps be described in terms of deviation from, or adherence to, generally known scripts, and levels of likelihood (probability measures) stored within these scripts. "Degree of deviation" can perhaps be determined by a calculation based on the presence or absence of familiar features, weighted for importance, whose function is to modify this stored probability. Before the episode, or sequence of episodes, can be matched against a script in this way, it must be generalised (since there cannot be a script for every conceivable event), and one is faced with familiar problems concerning which aspects are to be generalised. Consider the episode "Jack, while walking across the battlefield on his way home, came across a corpse lying on the ground". If the element "battlefield" is generalised to "place", the episode is likely to receive a much lower plausibility rating than if it is not.

But really, to use plausibility in this way is a short-cut which may well not correspond to the way in which human storytellers handle the problem - there is little evidence that humans have situations stored in their minds, indexed according to plausibility, or that they do plausibility calculations. When a human reader makes a judgement of likelihood in real life, or of *vraisemblance* in a story, they are making a comparison. On the one hand, they have the event, action, or whatever, that is under consideration. On the other, they have a collection of events or actions that they have stored in their memory at earlier points in their life, and their fiction-reading. Tversky & Kahneman (1973) have described such judgements in terms of the "heuristic of availability". By this they mean that when a subject is asked to judge the typicality of a case, they endeavour to remember a similar case, and use the ease with which such a case can be recalled as an indication of how frequently such cases occur and therefore how likely the current case

is. They have provided convincing evidence that this is indeed the heuristic that people typically employ in such a case (op.cit.). The heuristic might perhaps be modeled by a case-based reasoning system.

The concept of surprisingness is related to (though not identical to) the concept of implausibility. The author of a story – particularly a twist-centred story – will wish to devise episodes, or perhaps a series of episodes amounting to a plot, that are surprising but which pass a threshold of vraisemblance.

Ortony & Partridge (1987) have described surprisingness in terms of three different sorts of expectation failure. In each case, there is a conflict or inconsistency between the input proposition and something in the subject's mind: in the first form, it is an active prediction or expectation. In the second form, it is their knowledge or beliefs. In the third form, it is what might be judged to be normal or usual. We suspect the existence of another form of surprisingness: the idea that the subject searched for a particular solution to a problem, or interpretation of circumstances, but did not find it. When presented with this solution/interpretation, they are aware that they did not find it, but recognise the solution/interpretation to be effective or correct.

Macedo & Cardoso (2001) have proposed a model of creativity based on surprise value. In this model, surprise value is calculated as the degree of unexpectedness of an event: $1 - p(x)$, where $p(x)$ is the probability of the event, which in turn is calculated as the mean of the conditional probabilities of its component parts, which in turn are calculated using Bayes's formula. They use this value to guide a process of selecting a solution to a problem, the solution being represented as a subgraph needed to complete an existing graph. The solution is chosen from candidate solutions by applying a utility function, based partly on surprise value and partly on an appropriateness value. The effect is to generate a solution to some task – presumably a task in which creativeness is valued – which is both effective and original.

Macedo & Cardoso's approach (op.cit) does not employ the idea of surprisingness as the presentation of a solution that the subject searched for but did not find. It is almost certainly more tractable than an approach that did. It is hard to imagine that a computer system could model the subject's thought processes, in terms of searching for a solution or an interpretation, and predict what the subject would not find, and then generate this unlocatable solution itself.

This paper has described a theoretical model of story generation and a partial computational implementation of that model. The model generates two versions of a seed story, the overt and concealed versions. It then

generates a twist, so that the audience, when presented with the overt story and the twist, is forced to re-interpret the surface story as the concealed story.

As TWISTER is developed, it is intended that explicit methods for manipulating vraisemblance and surprisingness will be incorporated into it. We hope that TWISTER will become a system that generates interesting stories with twists, and a system that can help to explore the story generation process.

Acknowledgements

This work is supported by a part-time studentship from Middlesex University.

References

- Bringsjord, Selmer & Ferrucci, David (2000) *Artificial intelligence and literary creativity*. Lawrence Erlbaum Associates, Mahwah NJ.
- Correia, Alfred (1980). Computing story trees. *American Journal of Computational Linguistics*. 6, pp135-149.
- Fellbaum, Christiane (ed.) (1998) *WORDNET: an electronic lexical database*. MIT Press, Cambridge, MA.
- Graves, Robert (1955) *The Greek myths*. Penguin Books, Harmondsworth.
- Macedo, Luis & Cardoso, Amilcar (2001) Creativity and surprise. *AISB'01 symposium on artificial intelligence and creativity in arts and sciences*. 21-24 March, York, UK.
- Meehan, James. (1976) *The metanovel: writing stories by computer* (Tech. Rep. no.74, doctoral dissertation). Yale University, Dept. of Computer Science.
- Ortony, A & Partridge, D (1987) Surprisingness and expectation failure; what's the difference? *Proceedings of the 10th International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, Los Altos CA.
- Schank, Roger C. & Abelson, R. (1977) *Scripts, plans, goals, and understanding*. Lawrence Erlbaum Associates, Hillsdale NJ.
- Tansley, D. S. W. & Hayball, C. C. (1993) *Knowledge-based systems analysis and design*. Prentice Hall, Hemel Hempstead UK.

Turner, Scott R. (1994) *MINSTREL: A computer model of creativity and storytelling*. Doctoral dissertation, Artificial Intelligence Laboratory, Computer Science Department, University of California, Los Angeles.

Tversky, A & Kahneman, D (1973). Availability: a heuristic for judging frequency and probability. *Cognitive Psychology*, 5, pp207-232.

Automated Puzzle Generation

Simon Colton

Division of Informatics
University of Edinburgh
Edinburgh EH1 1HN
United Kingdom
simonco@dai.ed.ac.uk

Abstract

We give a characterisation of certain types of puzzle in terms of the structure of the question posed and the nature of the answer to the puzzle. Using this characterisation, we have extended the HR theory formation system (2) to enable it to automatically generate puzzles given background information about a set of objects of interest. The main technical difficulty to overcome was to ensure that the puzzles generated by HR had a single solution (up to a level of plausibility). We give details of the implementation and some results from its application.

1 Introduction

Broadly characterised as the generation of novel information about a domain given some background information, the specific tasks in machine learning include finding a concept given some positive and negative examples of the concept, and generating a scheme for predicting the nature of a given object. We can add to this list of tasks the generation of problems and puzzles about a given set of objects in a domain. While much effort has been expended on determining and implementing general problem solving techniques (8; 5), there has been little research on the question of automatically generating puzzles. This is in spite of the observations that scientific advance sometimes occurs through specifying exactly the right question to ask (and finding the answer), and that setting exercises is an important educational tool.

Our primary aim here is to determine the nature of certain types of puzzle and to show how the production of them can be automated. This is similar to how Ritchie specified the internal linguistic structure of a joke, separately to the notion of what makes a joke funny (7). However, a secondary aim is to determine the nature of a “good puzzle”. While a full study of the notion of a good puzzle would require a field test of the puzzles produced automatically, we make some preliminary observations and some plausible suggestions about increasing the difficulty of a puzzle. We begin in §2 with an analysis of some puzzles from which we make a characterisation of certain types of puzzle. In §3, we discuss ways of increasing the difficulty of the puzzles. In §4 we discuss the HR theory formation system (2) and in §5, we describe the extension to it which has enabled puzzle generation. We present some results in §6 and conclude in §7 with a brief discussion of puzzles in the context of creativity research.

2 A Characterisation of Puzzles

We take as examples for study eight puzzles taken verbatim from the web pages at queendom.com, a popular site for people interested in solving puzzles.

-
1. Q. Which is the odd one out:
coconuts, oysters, clams, eggs, walnuts, haddock?
A. Haddock: all the others have shells.
 2. Q. Which is the odd one out:
hair, triangles, squares, plants, words, trees?
A. Triangles: all the others have roots.
 3. Q. Which is the odd one out:
soft ice cream cone, salsa, landscape, raw vegetable, sales, chips?
A. Salsa: all the others can be dipped.
 4. Q. Jingle is to corporation as ___ is to politician:
(a) campaign (b) platform (c) slogan (d) promises?
A. Slogan.
 5. Q. Hair is to stubble as potatoes are to ___:
(a) french fries (b) sweet potatoes (c) potato skins (d) vegetable?
A. French fries.
 6. Q. What is next in the sequence: 3, 8, 15, 24, 35?
A. 48: Starting from 2, square each consecutive integer then subtract 1.
 7. Q. What is next in the sequence: 2, 7, 4, 14, 6?
A. 21: These are the multiples of 2, alternatively interspaced with the multiples of 7.
 8. Q. What is the next in the sequence: 15, 210, 115, 220, 125? A. 230. These are the multiples of five with the digit 1 or 2 alternatively placed in front of each number.
-

The first thing to notice about these puzzles is that they fall into three classes:

- Odd one out puzzles (puzzles 1, 2 and 3)
- Analogy puzzles (puzzles 4 and 5)
- Next in sequence puzzles (puzzles 6, 7 and 8)

Indeed, the puzzles at queendom.com are arranged into classes, and there are many others, such as “hidden word” and “word associations”.

We impose the following characterisation on these puzzles: each puzzle consists of (i) a question (ii) a set of choices, one of which is the answer (iii) the answer (iv) an explanation which consists of a single, positively stated concept. For example, in puzzle 1, the question is: “Which is the odd one out”, the set of choices is {coconuts, oysters, clams, eggs, walnuts, haddock}, the answer is haddock and the concept is the notion of having a shell.

We note that the puzzles stated above do not all fit this characterisation, and some flexibility is required. Firstly, the next in sequence puzzles do not have a set of possible answers to choose from. However, the implicit assumption is that the answer is an integer, hence we can say that the set of choices in this case is the set of natural numbers. Secondly, the analogy puzzles do not have an explanation. This is perhaps because, in the examples given above, the analogy is not exact, but rather the solution has a closer analogy than the other possible answers. However, it is reasonable to assume that puzzles where the analogy is via a single concept are acceptable. For instance, if the concept was “doubling” and the domain was numbers, then the analogy puzzle:

Q. 10 is to 20 as 30 is to 60 as 40 is to ____:
70, 80, 90?

A. 80 because 20 is 10 times 2, 60 is 30 times 2 and 80 is 40 times 2.

is valid. Hence to impose our characterisation, we assume that there is a single concept which explains the solution to the puzzle.

We call this concept the *embedded* concept, and by saying that it must be positively stated, we mean that the concept is not obviously the negation of another concept. For instance, in odd one out puzzles, it is usual for the odd one out to be lacking a property that the others all have. Puzzles where the opposite is the case are to some extent unsatisfying, for instance, consider the puzzle:

Q. Which is the odd one out: 2, 3, 9, 20?

A. 9 because it is a square and the others are not.

We see that the standard format has been violated: it is expected that the solution lies in finding a concept which 2, 3 and 20 satisfy and 9 doesn't, not in finding a property unique to 9. We also note that the embedded concepts are, in general, fairly simple and are rarely formed through disjunction. For example, if an explanation was something like: “because the others are either a fruit or

have a shell”, this too would be slightly unsatisfying.

Given this characterisation, we note that the three puzzle types differ only in the presentation of the question and explanation. In odd one out puzzles, the problem is to find the choice which does not have a property that the others have. In analogy puzzles, the problem is to find the choice which is most analogous with the given object. In next in sequence puzzles, the problem lies in extrapolating a sequence.

Another important observation is that the puzzles must have only one plausible solution. Consider, for example the puzzle:

Q. Which is the odd one out: 4, 9, 18, 36?

Here, there are (at least) two simple answers: (a) 18 is the odd one out because the other are all square numbers and (b) 9 is the odd one out because the others are all even numbers. Hence this puzzle is ineffective as there is a possibility that the solver will get the “wrong” answer, but not accept that his or her solution was worse than the supposed “right” answer. We see that the puzzle generator must take into account all simple concepts and ensure that the examples for the embedded concept do not form a puzzle for another embedded concept of similar or lesser complexity.

3 Increasing the Difficulty of Puzzles

We have shown that – with a little flexibility – the eight puzzles provided fit into a characterisation in terms of a question statement, a set of objects from a domain to choose from, an answer, and an embedded concept which explains the solution. This characterisation is useful in implementing puzzle generation, as discussed in §5 below. We have also noted that for each puzzle, the set of objects of interest do not embed a puzzle of similar or lesser complexity. This will help to ensure (but by no means guarantee) that the solver will be satisfied with the solution.

We must also concern ourselves with the difficulty of the puzzle. In particular, a puzzle which is too easy will be of little interest to the solver, yet a puzzle which is so difficult that no-one can find the solution is also ineffective. A balance needs to be found so that the intended puzzle solvers have some difficulty finding the answer, but in general, they eventually prevail. We describe below three properties of a puzzle which may help us to assess the difficulty of the puzzle. We make an implicit assumption that the time taken to solve a puzzle gives an indication of the difficulty of that puzzle.

3.1 Number of Choices

In the eight puzzles given above, the number of objects to choose the answer from ranges from 4 to 6, and increasing the number of objects is a potential way to increase

(or indeed decrease) the difficulty of the puzzle. A plausible method for solving odd one out puzzles is to choose an odd one out and attempt to find a property that the others share. Another possible method is to take two objects and find a property that they share, then try to add a third with the same property and so on, until all the objects have been added, with the exception of the odd one out. In either case, having more examples will slow down the solving process, and hence can be thought of as increasing the difficulty.

Similarly, to solve analogy puzzles, a plausible method is to choose a possible answer and attempt to find a concept which provides both an analogy between the pairs of objects stated in the question and an analogy between the final object stated in the question and the answer chosen. Again, increasing the number of objects to choose an answer from may increase the time taken to find a solution.

3.2 Complexity of the concept

Another possible way to increase the difficulty of the puzzle is to increase the complexity of the embedded concept, because, to search for the answer, there must be a search for the explanatory concept. Hence, making the solver search further for that concept will increase the difficulty of the puzzle.

We noted that, in general, we've observed that the concepts embedded in the puzzle are usually not particularly complex. One explanation for this could be that it is unlikely that a simpler solution will exist if the concept chosen to embed is simple. That is, checking for uniqueness of the solution up to a particular complexity level will be easier if the concept chosen to embed is itself simple. With this observation, we can tentatively state that, if the uniqueness-checking mechanism is reliable and efficient, then there is no reason why the difficulty of puzzles couldn't be raised by choosing complicated concepts to embed in the puzzles.

3.3 Disguising Concepts

If we look at puzzle 7 above, there is clearly an attempt to disguise the puzzle by involving another concept in the examples given in the problem statement. In that case, the embedded concept is multiples of 7, and the disguising concept is multiples of 2. It could be argued that multiples of 2 is also an embedded concept, and that our characterisation should take into account the possibility of multiple concepts providing the explanation. However, we note that the answer is only dependent on the concept of multiples of 7, and the explanation could have been provided as: "every other term is a multiple of 7, hence the next is 21", with no explicit reference to the concept of multiples of 2. Therefore, we call the secondary concept the disguising concept, and note that it only serves to increase the difficulty of the puzzle by diverting the solver's attention from the real problem in the puzzle.

It is important to note that there are a number of ways to employ disguising concepts, and it is part of the solving process to (a) realise that the puzzle is disguised and (b) work out and ignore the disguising concept(s). For number sequences, interleaving the disguising concept as in puzzle 7 is a common construction, as is appending the disguising sequence onto the front (or end) of the embedded sequence, for instance puzzle 8 above, where the embedded concept is "multiples of five", and the disguising concept is placing the number 1 or 2 alternatively in front of each number.

If we assume the strategies for finding the odd one out discussed in §3.1, then we can propose a way to use a disguising concept *C* in odd one out puzzles: choose examples which *all* share the property prescribed by *C*. In this way, as the solver attempts to find a property which the majority of examples share, he or she may come across *C*, because as all the examples share the property. As this cannot lead to a solution to the odd one out puzzle, it simply serves to slow the solver down. For instance, in puzzle 1 above, the solver might realise that the majority of the examples provided are foodstuffs, and try to find an example which is not. In this case, they would be disappointed to realise that *all* the examples are foodstuffs and therefore another concept is embedded in the puzzle.

4 The HR Program

The HR system performs automated theory formation by inventing concepts, making conjectures, proving theorems and finding counterexamples (2). The main functionality used for the application to puzzle generation is concept formation, which is achieved by using production rules which take one (or two) old concepts as input and output a new concept. The production rules perform as follows:

- **Compose**: this composes concepts using conjugation.
- **Disjunct**: this combines concepts using disjunction.
- **Exists**: this introduces existential quantification.
- **Forall**: this introduces universal quantification.
- **Match**: this equates variables in predicate definitions.
- **Negate**: this finds compliments by negating clauses.
- **Size**: this counts set sizes.
- **Split**: this instantiates variables.

As an example of how the production rules are employed, figure 1 shows how HR can construct the concepts of squares minus one (as used in puzzle 6 above). We see that the concept of square numbers is constructed from the user-given concept of multiplication using the match and exists production rules. The concept of adding 1 (equivalently subtracting 1) is constructed using the split rule to instantiate the number being added to 1. Finally, the compose rule is used to join these two concepts into the desired concept. We say that the concept produced in figure 1 is of *complexity* 5, because five concepts (including itself) are used in the construction path. The complexity

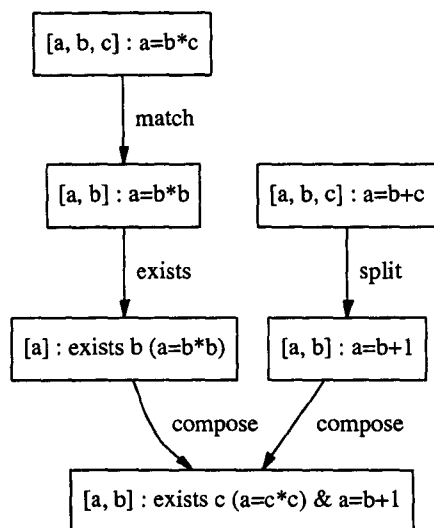


Figure 1: Example construction via production rules

of a concept gives some indication of how complicated the notion expressed in the concept is, and we use this in the puzzle generation, as discussed in §5 below.

Another fact used in puzzle generation is that each concept produces a categorisation of the objects of interest in a domain, for example the concept of squares minus one categorises the numbers 1 to 15 into two categories:

$[3, 8, 15], [1, 2, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14]$

The fact that HR knows when a concept is a specialisation of the objects of interest is also employed in puzzle generation. For example, once constructed, HR knows that the concept of squares minus one actually specialises the concept of integers. Furthermore, HR knows when one concept is a generalisation of another, information which is also used in the puzzle generation process. For instance, HR knows that the concept of prime numbers is a generalisation of the concept of odd prime numbers. HR also maintains knowledge of which concepts are functions. The user specifies which concepts supplied as background knowledge are functions, and HR propagates this information as the theory is built, i.e., it knows in which circumstances the application of a production rule will result in a function, and records this information. In particular, the size production rule (which counts sizes of sets) always produces a function. For more details of HR's concept formation, see (4), or chapter 6 of (2).

5 Extension for Puzzle Generation

Our original motivation for using HR for puzzle generation was the knowledge that HR can form a theory about a set of objects of interest, given some concepts about those objects. Hence it seemed that the task of producing puzzles with unique solutions could be achieved by (i) form-

embedding it in a puzzle in such a way that no other concept in the theory could be used to provide a solution to the puzzle. Hence, we extended HR to produce puzzles of the three types mentioned above (odd one out, analogy and next in sequence). There are many ways to produce such sequences, and we have only explored one or two avenues for each puzzle type – there is still much work needed to make HR proficient at generating puzzles.

Due to the characterisation afforded above, much of the implementation was not specific to the particular puzzle types. However, the two main areas where the implementation differs for the puzzle types is in checking that no other simple solution to a puzzle was possible, and in disguising the puzzle, and we deal with these in the subsections below. Also, various observations made in the characterisation above influenced our approach to automated puzzle generation, including:

- Negated concepts are not usually provided as the solution to puzzles. Based on this observation, we decided to omit the negation production rule when producing theories from which puzzles will be generated. This helps avoid puzzles where the explanation is a negated concept (for example, the odd one out is a square number, whereas the others are non-squares).
- Concepts with disjunction are not usually provided as the solution to puzzles. Based on this observation, we decided to omit the disjunct production rule when producing theories from which puzzles will be generated. This helps to avoid puzzles where the explanation concept has disjunction (for example, the embedded concept is “primes or squares”).
- The solution to a puzzle is supposed to be the simplest such solution. As mentioned above, the methods for checking uniqueness of solution is described for each puzzle type below. In essence, however, HR simply checks that there is no simpler solution to a puzzle it has generated by looking through all the concepts in the theory. Our notion of whether one solution is simpler than another was changed by some initial results from HR: we originally intended to use HR's complexity measure to determine which solution was simplest, with the least complex concept embedded being the simplest solution. However, we found that sometimes, more complex concepts were actually easier to understand than less complex ones. In particular, the match production rule will generate a concept with complexity one more than its parent, but in many cases, the definition of the concept produced will be simpler than that of the parent. Hence, we decided that, once all concepts up to a particular complexity limit (usually 4 or 5) had been generated, HR should discard puzzles which have two solutions, even if one embedded concept is more complex (according to HR's measure) than the other. While this lowers the yield of puzzles, it also reduces the possibility of a puzzle being generated with two or more plausible solutions.

5.1 Odd One Out Puzzles

After a theory containing many concepts has been formed by HR, the user decides the number of choices, n , and asks HR for odd one out puzzles. HR then looks at each specialisation concept, C , in turn and takes each subset of n objects of interest where the first $n - 1$ have the property prescribed by C and the last one does not have the property. For each tuple of n objects, it checks whether C is the only concept embedded in the tuples for an odd one out puzzle. For instance, if it took the integers 3, 5, 17, 8 to embed the concept of prime numbers (true for 3, 5 and 17, not true for 8), it would run through all the other specialisation concepts and check that no other solution was possible. In this case, when it came to the concept of integers with one digit, it would realise that 17 could be considered the odd one out instead of 8, because 17 has two digits, whereas the others have only one. Hence this quadruple would be rejected for an odd one out puzzle where the concept to be discovered is prime numbers.

Tuples of objects for embedding a concept C into a puzzle are rejected if another concept D can be embedded as the solution to the puzzle, with one exception: when C is a generalisation of D . For example, if 3, 5, 17, 22 were chosen to embed the concept of prime numbers in a puzzle, then the concept of odd prime numbers could be used to reject this tuple, as it provides another answer (22 is not an odd prime number, the others are). However, as mentioned above, HR keeps track of which concepts are specialisations of another and it knows that the concept of prime numbers is a generalisation of odd prime numbers, so it will not reject the tuple for the puzzle. In effect, the solver is expected to be happy with the answer of prime numbers for the puzzle, if they have decided that the answer was odd prime numbers, because the former is a simpler solution. Hence, as HR knows which concepts are generalisations of the others, it can avoid rejecting a tuple when a more specialised concept can be embedded in the objects chosen.

For each concept, HR collects all the tuples which pass the uniqueness test, and determines the level of disguise of each, measured as the number of concepts in the theory which specify a property that *all* the objects satisfy. For instance, if the concept of square numbers was embedded with the examples 4, 16, 20, 36, then the concept of even numbers would be a disguising concept for this puzzle, as all the numbers are even. The choice of objects determines the number of other concepts in the theory which act as disguising concepts. We found that, even with a high level of disguise, puzzles with simple concepts were easy to solve. Hence, we used an interestingness measure which takes the average of the complexity of the embedded concept and the level of disguise. For each concept, the most interesting puzzle is taken as a representative puzzle for that concept. To finish the puzzle, HR randomly re-orders the choices, so that the odd one out is no longer the last choice.

5.2 Analogy Puzzles

Our treatment of analogy puzzles has been fairly shallow so far. After a theory has been formed, HR takes each specialisation concept and chooses objects of interest to embed the concept in a puzzle. There are many ways in which the analogy could be set up, and we have so far only experimented with one: HR produces puzzles of the form "A is to B as C is to ?", where objects A, B, C and the solution, all share the property expressed by the concept.

The user specifies the number, n , of choices which will be available in the puzzle, and then to generate puzzles from a concept C , HR takes each category, T , in the categorisation of the objects of interest afforded by C , and counts the number of objects in T . If this number is four or more, then HR randomly chooses four objects O_1, O_2, O_3 and O_4 from T . From the objects of interest which are not members of T , HR chooses objects X_1, \dots, X_{n-1} as the choices for the puzzle. It then adds O_4 to this list in a random position, so that O_4 becomes X_i for some i and there are n choices for the answer to the puzzle. The puzzle is then stated as: " O_1 is to O_2 as O_3 is to X_1, X_2, \dots, X_n ?"

As with odd one out puzzles, for each puzzle HR generates, it checks whether there is another equally plausible solution. In the case of analogy puzzles, to do this, it runs through every other concept D , and checks whether there is a category V in the categorisation produced by D such that V contains O_1, O_2 and O_3 and V also contains a single object from the set X_1, \dots, X_n . In such cases, D also provides a solution to the puzzle, so the puzzle should be discarded. As with odd one out puzzles, the exception to this rule is when D is a specialisation of C .

For analogy puzzles, we do not yet worry about disguising the puzzle. For this reason, for each concept to embed in a puzzle, HR stops once it has found a puzzle which passes the uniqueness test. To stop biasing towards certain objects of interest, the category T is chosen randomly from all the categories, and $\{O_1, O_2, O_3, O_4\}$ and $\{X_1, \dots, X_{n-1}\}$ are chosen randomly from all possible sets until the supply is exhausted.

5.3 Next in Sequence Puzzles

Three common formats for next in sequence puzzles are:

- The embedded concept is a number type and successive examples of the number type are given in order, with the next such number being the answer, e.g., puzzle 10 in §2 has the concept of multiples of five embedded.
- The embedded concept is a function f mapping the integers to themselves, and the function is applied to successive integers $n, n + 1, \dots, n + k$ for some n and k , with the answer being $f(n + k + 1)$. For example, in puzzle 6 in §2, the function is $f(n) = n^2 - 1$ and the application of f starts with $f(2) = 3$ and continues until $f(6) = 35$, so that $f(7) = 48$ is the answer.

- The embedded concept is again a function f , but it is applied recursively to the previous terms in the sequence. An example of this is the Fibonacci sequence: 1, 1, 2, 3, 5, 8, ..., where the n th term in the sequence is gained by adding together terms $n - 1$ and $n - 2$ (with 1 and 1 chosen arbitrarily as the first two terms of the sequence).

At present, HR only has the functionality to produce puzzles of the first and second format, although we do not envisage any problems extending HR's range to include the third format.

To generate puzzles of the first type, HR forms a theory in the domain where the objects of interest are integers and then looks for concepts C which are number types, i.e., specialisation concepts. Given that the user has specified that n integers should be given as clues to the puzzle, HR chooses n successive integers from the integers which have the property prescribed by C . Instead of choosing the starting point for the clues randomly, we made HR choose them in such a way that the clue numbers were as large as possible. For example, given the numbers 1 to 30 about which to form a theory, when producing a puzzle with 5 integers as clues to a puzzle with prime numbers as the embedded concept, HR would choose the numbers 11, 13, 17, 19 and 23, with the final prime number that HR knows about (29) being held back as the answer. We found that many sequences have similar numbers early on in the number line, and hoped that, by choosing the largest numbers possible, it would be easier to ensure the uniqueness of the solution and also perhaps increase the difficulty of the puzzle.

To generate puzzles of the second type, HR takes each concept C , which it knows to be a function, f , mapping the integers to themselves. HR then chooses a number x to start at, and provides the integers $f(x)$, $f(x + 1)$, ..., $f(x + n)$ as clues to the puzzle, with $f(x + n + 1)$ held back as the answer. As with the first type, we made HR choose x to be as high as possible. With both types of sequence, HR checks that the solution is unique for all the concepts in the theory. It takes into account the fact that an alternative solution could come from a sequence generated by number types or by a function.

Rather than trying to find puzzles with the most disguise – as is the case with odd one out puzzles – HR explicitly imposes disguise on a puzzle by interleaving the clues with another sequence of similar complexity. HR does this only if the complexity of the embedded concept is very low (the user sets this, usually at complexity 2 or less). This means that only the puzzles about very simple concepts such as even numbers are disguised. For instance, HR disguises the concept of numbers with the digit 2 in them (given with examples 12, 20, 21) by interleaving them with the concept of even numbers (2, 4, 6, 8) to produce the puzzle: “what is the next in the sequence: 2, 12, 4, 20, 6, 21, 8?”

We chose the starting point for the disguising sequence to be as low as possible. Our rationale behind this was that

we wanted the solver to recognise the disguised concept (thus being somehow distracted by it). After disguising, HR again checks for the uniqueness of the solution, and will backtrack over (i) the choice of start point for the disguising sequence (ii) the choice of disguising sequence and (iii) the start point for the embedded sequence, until it finds a puzzle which has a unique solution.

There are many other methods for generating and disguising next in sequence puzzles which we have not yet equipped HR with. HR will need to both generate puzzles using these methods and take them into account when checking for uniqueness of the solution. For instance, HR will eventually take into account that many sequence extrapolation puzzles are solved by taking the difference sequence, i.e., the difference between successive terms.

6 Results

Unfortunately, time has not permitted the kind of extensive development and testing of HR's puzzle generation capabilities that we wanted to report on here. In particular, we wanted to use HR to generate puzzles of a visio-spatial nature, but we have not had time yet to do this.

6.1 Animals Puzzles

To generate odd one out and analogy puzzles, we chose the animals dataset which is supplied not with HR, but rather with the Progol machine learning program distribution (6). This dataset consists of 18 animals described with 12 properties, such as whether or not they produce eggs, which habitats they dwell on (air, water, land) and so on. We set HR's complexity limit to be five and ran the search to exhaustion using the compose, exists, forall, match, size and split production rules. It took 65 seconds to build the theory on a Pentium 500Mhz processor. We then asked for all odd one out and analogy puzzles with four choices which could be generated as prescribed in §5 above. This took a further 7 seconds and produced 31 puzzles, which we supply in appendix A in such a way that the choices for the answer are given, with the correct one bearing an asterisk, and the embedded concept and disguising concepts stated below the choices.

With some puzzles, the complexity of the embedded concept may have made the puzzle of sufficient difficulty to be interesting, for example, the solver needed to know in puzzle 25 that of the four animals supplied, dolphin was the only one to have a single habitat (water). With other puzzles, however, the choice of the animals negated the complexity of the concept, e.g., puzzle 27 asked: dog is to cat as eagle is to (a) lizard (b) eel (c) ostrich (d) trout, with ostrich being the only plausible solution, regardless of the concept which related dog, cat, eagle and ostrich (which was being a homeothermic land-dweller).

Puzzle 7 had the most disguise: the objects to choose the odd one out from were ostrich, platypus, eagle and

penguin, which share three properties; they are homeothermic, produce eggs and have two legs. However, as they are also all animals (a disguising concept for all the odd one out puzzles), and because HR invented the concept of being homeothermic and producing eggs, which they all share, this puzzle scored 5 for disguise. This suggests that an improvement in measuring disguise would be to make sure that the properties the objects share are not really being counted twice. In general, we were disappointed with the level of disguise, and it was at such a low level to be unlikely to have an effect on the solver's ability to find the solution.

An important observation is that relatively few concepts used in the puzzles were of complexity 4 and 5. This is partly due to there being slightly fewer such concepts in the theory, but also because the more complex concepts tend to be more specialised. As the complexity (and hence specialisation) of a concept increases, it becomes less likely that the concept will have sufficient positive examples to make a puzzle out of it. For instance, the concept of birds which fly has only one example in the animals dataset: eagle. Hence this concept cannot be used in an odd one out puzzle. The solution to this, of course, is to provide more objects of interest (in this case, more animals) in the background information, although this will mean that the theory formation and puzzle generation will take more time.

To summarise the animals results, as the level of both disguise and complexity of the puzzles was in general fairly low, we expect these puzzles to be easily solved. It appears, however, that a sizeable proportion of the puzzles would be satisfying to the solver, in that they would probably come up with the solution that HR prescribed.

6.2 Integer Sequences

To produce next in the sequence puzzles, we ran HR in number theory with the numbers 1 to 30 and the concepts of: multiplication, addition, divisors and digits, which are all commonly used concepts in sequence extrapolation puzzles. Due to experience in this domain, to reduce the complexity of the concepts generated, we used a complexity limit of 4 and did not use the forall production rule, using only compose, exists, match, size and split. Again, we ran the exhaustive search to completion, and then asked for all next in sequence puzzles where 6 numbers were given as clues. It took 165 seconds on a Pentium 500Mhz processor to form the theory and a further 2 seconds to generate the 24 puzzles given in appendix B.

Some of the puzzles generated are promising, for instance HR made a puzzle about prime numbers (puzzle 4), which is a mainstay of human produced puzzles. Also, puzzle 10 asked: what is next in the sequence: 21, 22, 24, 25, 26, 28, with 30 being the answer, as this is the sequence of integers where there is a digit which divides the number. This is perhaps at around the limit of difficulty for a next in sequence puzzle. Also, the disguising pro-

cess for the simpler concepts seemed to work fairly well. In particular, in puzzle 3, HR used the same concept (multiples of 3) to disguise the embedded concept to produce the sequence: 21, 3, 24, 6, 27, 9, which was interesting.

However, many of the puzzles highlight problems with HR's puzzle generation which will need improvement. In particular, some of the concepts, while only at complexity 4 in HR's terms might be a little complicated to reasonably expect the solver to identify them. These include the number of even divisors of the integers (puzzle 20). The situation is exasperated by the policy of starting the puzzles as far down the number line as possible. While this is a good idea for puzzles involving number types, it seems likely that expecting the solver to realise that the sequence 6, 0, 2, 0, 4, 0 has been generated by writing down the number of even divisors of 24, 25, 26, 27, 28 and 29 is asking too much. This suggests that for sequences generated by applying a function to successive integers, the clues to the puzzle should start by applying the function as close to the number 1 as possible.

The last two puzzles in appendix B highlight the fact that puzzles with a simpler solution can sometimes slip through. It is unlikely that the solver would accept HR's convoluted answer for the next in the sequence: 0, 1, 2, 3, 4, 5 (puzzle 24). In this particular case, as HR was given the numbers 1 to 30 to work with, not 0 to 30, it never realised that the sequence 0, 1, 2, 3, 4, 5 was part of the natural number sequence. Similarly, HR should have realised that the sequence 11, 12, 12, 13, 13, 14 (puzzle 23) could be generated by repeating the natural numbers.

To summarise the next in sequence results from this session, it seems that the majority of the sequences generated from number types produced acceptable puzzles and the policy of taking the largest numbers possible worked well. However, the policy worked badly for the sequences generated from functions and this combined with the high complexity of the embedded concepts meant that the puzzles were perhaps too difficult.

7 Conclusions and Further Work

The generation of puzzles is a machine learning task which, to our knowledge, has rarely been tackled. To implement puzzle generation, we identified three common types of puzzle and found a characterisation to fit the three types. This enabled us to extend the HR program to generate odd one out, next in sequence and analogy puzzles. We presented results in two domains to highlight the strengths and weaknesses of our approach. We found that the main technical problem with puzzle generation was ensuring the uniqueness of the concept supposed to explain the puzzle solution. We used automated theory formation to build a theory so that, up to a reasonable level of complexity, HR could check that there was no other simple solution to a puzzle. While this approach is not perfect – indeed it is a very difficult problem to predict the cre-

ative approaches to problem solving that people will employ – we found that in most cases the solution found by the solver was likely to be the one they were supposed to find. We argue in (3) that the theory formation approach employed by HR is perhaps better suited for puzzle generation than other machine learning techniques which aim to find a single concept.

Puzzle generation, which, in its broadest sense encompasses many activities ranging from identifying the correct question to propagate research to the setting of exercises to educate students, is an intelligent activity which itself requires creativity, but which must take into account the perceived creativity of the intended puzzle solvers. In particular, a puzzle must be generated in such a way that the given answer is more plausible than any other the solver might think of. In terms of Boden's characterisation of creative acts (1), a puzzle setter must choose examples for the puzzle which make the solver act P-creatively (produce a solution which is a personal discovery), rather than H-creatively (produce a solution which is historically novel). This is because, by definition, a H-creative solution to a puzzle will be wrong (in the sense that it was not the answer intended by the puzzle setter). Once we have improved the quality of HR's puzzles and tested them with human solvers as mentioned below, we hope to have a greater insight into the nature of machine/human creativity with respect to puzzle generation and solving.

The results from the initial testing of HR's puzzle generation show that there is still much more to be done in order to increase the quality of the puzzles it produces. We hope to follow two main directions with this work. Firstly, we must test whether human puzzle solvers find the puzzles HR produces interesting, and we hope to use a web-based experiment which asks the solvers to rate (a) how difficult they thought the puzzle was and (b) how happy they were with the solution. We will then compare their difficulty ratings with those used by HR, and attempt to reduce the number of puzzles produced for which they gave a different (but valid) answer.

Secondly, we hope to add a layer of sophistication to the assessment of difficulty by getting HR to model the puzzle solver as well as the puzzler setter. HR has already been applied to integer sequence extrapolation (4), and we hope to combine this application with the application to puzzle generation, so that one copy of HR generates the puzzles and another copy attempts to solve them, giving an indication of the difficulty in the process.

Acknowledgments

This work is supported by EPSRC grant GR/M98012. The author is also affiliated to the Dept. of Computer Science at the University of York. This research was inspired by a very interesting meeting with Prof. Herbert Simon, whose comments and questions greatly improved the author's understanding of puzzle solving and generation.

References

- [1] M Boden. *The Creative Mind*. Weidenfeld and Nicolson, 1990.
- [2] S Colton. *Automated Theory Formation in Pure Mathematics*. PhD thesis, Department of Artificial Intelligence, University of Edinburgh, 2000.
- [3] S Colton. An application-based comparison of automated theory formation and inductive logic programming. *Linköping Electronic Articles in Computer and Information Science (special issue: Proceedings of Machine Intelligence 17)*, forthcoming.
- [4] S Colton, A Bundy, and T Walsh. Automatic identification of mathematical concepts. In *Machine Learning: Proceedings of the 17th International Conference*, 2000.
- [5] Marsha J. Ekstrom Meredith. *Seek-Whence: A Model of Pattern Perception*. PhD thesis, Department of Computer Science, Indiana University, 1987.
- [6] S Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13:245–286, 1995.
- [7] G Ritchie. Describing verbally expressed humour. In *Proceedings of the AISB-00 Symposium on Creative & Cultural Aspects and Applications of AI & Cognitive Science*, 2000.
- [8] H Simon and A Newell. Heuristic problem solving: The next advance in operations research. *Operations Research*, 6(1), 1958.

A. Puzzles about Animals

1. Which is the odd one out?
i. eagle* ii. bat iii. cat iv. dolphin
answer: a produces milk
Disguising Concepts:
a is homeothermic
interestingness=1.5, disguise=2.0, complexity=1

2. bat is to cat as dog is to:
i. platypus* ii. lizard iii. trout iv. herring
answer: a produces milk
complexity=1

3. Which is the odd one out?
i. eagle ii. eel* iii. bat iv. dog
answer: a is homeothermic
interestingness=1.0, disguise=1.0, complexity=1

4. bat is to cat as dog is to:
i. shark ii. dragon iii. eel iv. dolphin*
answer: a is homeothermic
complexity=1

5. Which is the odd one out?
i. bat* ii. snake iii. lizard iv. penguin
answer: a produces eggs
interestingness=1.0, disguise=1.0, complexity=1

i. dog ii. t_rex* iii. dolphin iv. bat
 answer: a produces eggs
 complexity=1

 7. Which is the odd one out?
 i. ostrich ii. platypus* iii. eagle iv. penguin
 answer: a is a bird
 Disguising Concepts:
 a is homeothermic
 a produces eggs
 a has 2 legs
 a is homeothermic & a produces eggs
 interestingness=3.5, disguise=5.0, complexity=2

 8. Which is the odd one out?
 i. snake ii. herring* iii. lizard iv. dragon
 answer: a is a reptile
 Disguising Concepts:
 a produces eggs
 a is covered by scales
 interestingness=2.5, disguise=3.0, complexity=2

 9. crocodile is to dragon as lizard is to:
 i. penguin ii. platypus iii. bat iv. turtle*
 answer: a is a reptile
 complexity=2

 10. Which is the odd one out?
 i. cat ii. platypus iii. bat iv. dolphin*
 answer: a is covered by hair
 Disguising Concepts:
 a produces milk
 a is homeothermic
 interestingness=2.5, disguise=3.0, complexity=2

 11. Which is the odd one out?
 i. lizard ii. eagle* iii. dragon iv. herring
 answer: a is covered by scales
 Disguising Concepts:
 a produces eggs
 interestingness=2.0, disguise=2.0, complexity=2

 12. crocodile is to dragon as herring is to:
 i. penguin ii. eel iii. turtle* iv. bat
 answer: a is covered by scales
 complexity=2

 13. Which is the odd one out?
 i. snake ii. shark iii. bat* iv. dolphin
 answer: a has 0 legs
 interestingness=1.5, disguise=1.0, complexity=2

 14. herring is to shark as snake is to:
 i. trout* ii. platypus iii. turtle iv. eagle
 answer: a has 0 legs
 complexity=2

 15. Which is the odd one out?
 i. penguin ii. ostrich iii. cat* iv. bat
 answer: a has 2 legs
 Disguising Concepts:
 a is homeothermic
 interestingness=2.0, disguise=2.0, complexity=2

 16. bat is to eagle as ostrich is to:
 i. snake ii. platypus* iii. herring iv. dragon
 answer: a has 2 legs
 complexity=2

 17. Which is the odd one out?
 i. dog ii. dragon iii. cat iv. eel*

interestingness=1.5, disguise=1.0, complexity=2

 18. cat is to crocodile as dragon is to:
 i. snake ii. penguin iii. dolphin iv. t_rex*
 answer: a has 4 legs
 complexity=2

 19. Which is the odd one out?
 i. lizard ii. snake iii. dog iv. bat*
 answer: a lives in land
 interestingness=1.5, disguise=1.0, complexity=2

 20. cat is to crocodile as dog is to:
 i. shark ii. penguin iii. herring iv. ostrich*
 answer: a lives in land
 complexity=2

 21. Which is the odd one out?
 i. crocodile ii. turtle iii. bat* iv. dolphin
 answer: a lives in water
 interestingness=1.5, disguise=1.0, complexity=2

 22. crocodile is to dolphin as eel is to:
 i. dragon ii. ostrich iii. t_rex iv. trout*
 answer: a lives in water
 complexity=2

 23. eagle is to ostrich as penguin is to:
 i. t_rex ii. platypus* iii. dog iv. trout
 answer: a is homeothermic & a produces eggs
 complexity=3

 24. eel is to herring as snake is to:
 i. dragon ii. platypus iii. t_rex iv. trout*
 answer: a produces eggs & a has 0 legs
 complexity=4

 25. Which is the odd one out?
 i. dragon ii. bat iii. crocodile iv. dolphin*
 answer: 2={b: b is a habitat & a lives in b}|
 interestingness=2.0, disguise=1.0, complexity=3

 26. bat is to crocodile as dragon is to:
 i. cat ii. eel iii. eagle* iv. dog
 answer: 2={b: b is a habitat & a lives in b}|
 complexity=3

 27. cat is to dog as eagle is to:
 i. lizard ii. eel iii. ostrich* iv. trout
 answer: a is homeothermic & a lives in land
 complexity=4

 28. eagle is to ostrich as snake is to:
 i. cat ii. turtle iii. penguin iv. t_rex*
 answer: a produces eggs & a lives in land
 complexity=4

 29. crocodile is to lizard as snake is to:
 i. shark ii. eel iii. bat iv. t_rex*
 answer: a is a reptile & a lives in land
 complexity=5

 30. cat is to dog as lizard is to:
 i. t_rex* ii. eel iii. platypus iv. bat
 answer: a has 4 legs & a lives in land
 complexity=5

 31. eel is to platypus as shark is to:
 i. snake ii. eagle iii. turtle* iv. lizard
 answer: a produces eggs & a lives in water
 complexity=4

B. Next in Sequence Puzzles

1. What's next in the sequence:

27 2 28 12 29 20 (30)?

answer: a is an integer

Disguising Concepts:

2 is a digit of a

disguise=1.0, complexity=1

2. What's next in the sequence:

4 3 6 6 2 9 (8)?

answer: $b = \{c: c|a\}$

Starting with n=27

Disguising Concepts:

3|a

disguise=1.0, complexity=2

3. What's next in the sequence:

21 3 24 6 27 9 (30)?

answer: 3|a

Disguising Concepts:

3|a

disguise=1.0, complexity=2

4. What's next in the sequence:

7 11 13 17 19 23 (29)?

answer: $2 = \{b: b|a\}$

disguise=0.0, complexity=3

5. What's next in the sequence:

2 8 4 4 6 2 (8)?

answer: $b = \{c: c|a\} \& 2|b$

Starting with n=19

disguise=0.0, complexity=4

6. What's next in the sequence:

6 6 4 8 4 6 (8)?

answer: $b = \{c: c|a\} \& 2|a$

Starting with n=9

disguise=0.0, complexity=4

7. What's next in the sequence:

6 4 6 4 8 4 (8)?

answer: $b = \{c: c|a\} \& 3|a$

Starting with n=4

disguise=0.0, complexity=4

8. What's next in the sequence:

5 6 7 8 9 11 (22)?

answer: $1 = \{b: b \text{ is a digit of } a\}$

disguise=0.0, complexity=3

9. What's next in the sequence:

19 20 21 23 24 25 (26)?

answer: $2 = \{b: b \text{ is a digit of } a\}$

disguise=0.0, complexity=3

10. What's next in the sequence:

21 22 24 25 26 28 (30)?

answer: exists b (b|a & b is a digit of a)

disguise=0.0, complexity=4

11. What's next in the sequence:

2 1 1 0 1 0 (1)?

answer: $b = \{c: c|a \& c \text{ is a digit of } a\}$

Starting with n=24

disguise=0.0, complexity=4

12. What's next in the sequence:

3 1 2 1 2 1 (1)?

answer: $b = \{(c d): c*d=a \& d|c\}$

Starting with n=24

disguise=0.0, complexity=4

13. What's next in the sequence:

16 17 18 19 21 24 (25)?

answer: exists b c (b*c=a & c is a digit of b)

disguise=0.0, complexity=4

14. What's next in the sequence:

2 1 1 1 0 1 (0)?

answer: $b = \{(c d): c*d=a \& d \text{ is a digit of } c\}$

Starting with n=24

disguise=0.0, complexity=4

15. What's next in the sequence:

7 2 3 3 5 1 (7)?

answer: $b = \{(c d): c+d=a \& d|c\}$

Starting with n=24

disguise=0.0, complexity=4

16. What's next in the sequence:

20 22 23 24 25 26 (27)?

answer: exists b c (b+c=a & c is a digit of b)

disguise=0.0, complexity=4

17. What's next in the sequence:

2 1 3 1 3 1 (2)?

answer: $b = \{(c d): c+d=a \& d \text{ is a digit of } c\}$

Starting with n=24

disguise=0.0, complexity=4

18. What's next in the sequence:

0 1 1 2 2 2 (2)?

answer: $b = \{(c d): c+d=a \& d \text{ is a digit of } a\}$

Starting with n=24

disguise=0.0, complexity=4

19. What's next in the sequence:

2 5 2 2 2 3 (2)?

answer: $b = \{c: c|a\} \& 2 = \{d: d|b\}$

Starting with n=8

disguise=0.0, complexity=4

20. What's next in the sequence:

6 0 2 0 4 0 (4)?

answer: $b = \{c: c|a \& 2|c\}$

Starting with n=24

disguise=0.0, complexity=4

21. What's next in the sequence:

4 0 0 3 0 0 (4)?

answer: $b = \{c: c|a \& 3|c\}$

Starting with n=24

disguise=0.0, complexity=4

22. What's next in the sequence:

11 12 20 21 23 24 (25)?

answer: exists b (b is a digit of a &

$b = \{c: c \text{ is a digit of } a\})$

disguise=0.0, complexity=4

23. What's next in the sequence:

11 12 12 13 13 14 (14)?

answer: $b = \{(c d): c+d=a \& \text{exists } e (e+d=c)\}$

Starting with n=24

disguise=0.0, complexity=4

24. What's next in the sequence:

0 1 2 3 4 5 (6)?

answer: $b = \{(c d): c+d=a \& \text{exists } e f (e+f=d)\}$

Starting with n=24

disguise=0.0, complexity=4