Goldsmiths
UNIVERSITY OF LONDON

Proceedings of the 4th International Joint Workshop on

Computational Creativity

Amílcar Cardoso & Geraint A. Wiggins (editors)

17-19 June 2007

Foreword

The International Joint Workshop on Computational Creativity began life as two independent workshop series: the Creative Systems Workshops and the AISB Symposia on AI and Creativity in the Arts and Sciences. The two series merged in 2004, when the 1st IJWCC was held in Madrid, as a satellite workshop of the European Conference on Case Based Reasoning. Since then, two further satellite worshops have been held, at the International Joint Conference on Artificial Intelligence in Edinburgh, in 2005, and at the European Conference on Artificial Intelligence in Riva del Garda, in 2006.

This workshop constitutes the workshop's first attempts at independent existence, and the quality of the papers submitted suggests that the time is now ripe. This workshop received 27 submissions, all of which were subjected to rigorous peer review (at least 3 reviewers to each paper), and 17 full papers and 3 posters were accepted (one poster was subsequently withdrawn).

We believe this volume represents a coming of age of the field of computational creativity. It contains evidence of not only improvements in the state of the art in creative systems, but also of deep thinking about methodology and philosophy. An exciting new development is the inclusion, for the first time, of a session on applied creative systems, demonstrating that the field is now ready and able to impinge on broader artificial intelligence and cognitive science research.

As co-chairs, we would like to thank the programme committee and reviewers, our able local assistants, Ollie Bown and Marcus Pearce, and all those who submitted papers to make this a really exciting event.

Amíilcar Cardoso & Geraint A. Wiggins
Workshop co-chairs

Programme Committee and Reviewers

Barnden, John A – University of Birmingham
Bown, Oliver – Goldsmiths, University of London
Brown, David – Worcester Polytechnic Institute
Cardoso, Amilcar – University Of Coimbra
Collomosse, John – University of Bath
Colton, Simon – Imperial College London
Gervás, Pablo – Universidad Complutense de Madrid
Gomes, Paulo – University of Coimbra
Keller, Robert – Harvey Mudd College
Leite, João – New University of Lisbon
López, Jesús – University of Southern Queensland
López de Màntaras, Ramon – IIIA-CSIC
Machado, Penousal – University of Coimbra
Magnani, Lorenzo – University of Pavia
Moffat, David C. – Glasgow Caledonian University
O'Donoghue, Diarmuid P. – NUI Maynooth University
Pearce, Marcus T. – Goldsmiths, University of London
Pease, Alison – University of Edinburgh
Pereira, Francisco Câmara – University of Coimbra
Pérez y Pérez, Rafael – Autonomous Metropolitan University of Mexico
Rauchas, Sarah – Goldsmiths, University of London
Ritchie, Graeme – University of Aberdeen
Saunders, Rob – University of Sydney
Seco, Nuno – University of Coimbra
Stock, Oliviero – ITC-irst
Veale, Tony – University College, Dublin
Widmer, Gerhard – University of Linz
Wiggins, Geraint A. – Goldsmiths, University of London

Contents

Keynote

Text representation of music: from word processing to rule-based composition/improvisation

Bernard Bel, University of Aix-Marseille

The *Bol Processor* project originated in 1980 as a word processor facilitating the transcription of quasi-onomatopoeic syllables used as an oral notation system for Indian drumming. It grew up as an expert system (BP1) mimicking the ability to compose variations on a musical theme or assess their acceptability. *Pattern grammars* (a subset of type-2 formal grammars) proved appropriate for modelling the musical system under study. A stochastic learning device was implemented to infer weights from sets of examples accepted by the grammar, with the effect of enhancing the aesthetic quality of productions. None the less, field work revealed limitations inherent to the expert system approach when it comes to modelling sophisticated human improvisation skills.

In 1989 a numeric-symbolic learning device (QAVAID) was implemented in Prolog II for inferring grammars from examples. However, it has never been used in fieldwork because of its slow operation on portable computers of that time.

The next implementation of *Bol Processor* (BP2) addressed the issue of music composition and improvisation in the *MIDI* and *Csound* environments of electronic music. A new challenge was to deal with superimposed sequences of events (polyphony) within the framework of text-oriented rewriting systems. This was achieved by means of *polymetric representation*. Minimal descriptions of polyphonic/polyrhythmic structures may be "expanded" by the system to produce arbitrarily complex musical scores. This representation makes it possible to produce sophisticated time-patterns from information comprehensively imbedded in compositional rules, thereby maintaining the consistency of interpretation. This is a major discovery for computer music, as "human-like" phrasing is no longer achieved by randomness nor "interpretation rules".

Producing the actual performance requires additional information which the *Bol Processor* encapsulates in metrical/topological properties of "sound-object prototypes". A time-setting algorithm modifies sound-objects taking into account physical timing and their adjacent sound-objects, much in a similar way human speakers modify the articulatory properties of speech sounds with respect to the speaking rate and influence of adjacent segments (coarticulation).

Many composers and music teachers support the *Bol Processor* approach because of its underlying paradigm of text representation, i.e. "composing with pen and paper". It found its way long before the invention of markup languages, at a time only graphic interfaces were expected to capture the sophistication of compositional processes.

BP2 is currently implemented for MacOS 9 and MacOS X. The project has been open-sourced by Sourceforge at `http://sourceforge.net/projects/bolprocessor/` with the help of Anthony Kozar.

Bernard Bel is a computer scientist with background in electronics. In 1979 he started collaborating with anthropologists, musicologists and musicians on a scientific study of North Indian melodic and rhythmic systems. In 1981 he built the first accurate real-time melodic movement analyser (MMA) for the analysis of raga music. In 1986 he joined the French National Centre for Scientific Research (CNRS) in Marseille to continue a research on the rule-based modelling of training methods in traditional Indian drumming. He studied artificial intelligence under Alain Colmerauer and graduated with a PhD in theoretical computer science in 1990. Between 1994 an 1998, Bel was deputed to CENTRE DE SCIENCES HUMAINES (CSH, New Delhi) to carry on projects in the fields of computational musicology and social-cultural anthropology. He displaced his focus to "innovative" music forms: different ways of associating musical experience with information technology, and questioning the usual modernity/tradition dichotomy outside Western urban culture. In 1998 he joined LABORATOIRE PAROLE ET LANGAGE (CNRS, Aix-en-Provence) as a member of a team specialised in speech prosody and formal representations of language. Together with colleagues at CNRS he created the Speech Prosody Special Interest Group (SproSIG) under the banner of the International Speech Communication Association (ISCA).

Session 1
Creativity in Narrative

A Computer Model that Generates Biography-like Narratives

Samer Hassan[1], **Carlos León**[2], **Pablo Gervás**[3], **Raquel Hervás**[4]
Universidad Complutense de Madrid
Departamento de Sistemas Informáticos y Programación
[1]samer@fdi.ucm.es,[2]cleon@fdi.ucm.es,
[3]pgervas@sip.ucm.es,[4]raquelhb@fdi.ucm.es

Abstract

This paper presents an initial decomposition of the process of creative storytelling into subtasks that are relevant for studying where and how creativity plays a role from a computational point of view. Five basic subtasks are identified: building a world to act as setting for the story (including characters, locations, possible actions), generating a set of events that take place in that world, selecting from that set of events those that are worth telling, identifying a particular sequence in which to tell them, and finding appropriate linguistic realizations for each event in that sequence. To test the model, an initial prototype is presented that operates on logs generated artificially by a social simulation built by a multiagent system. A second module addresses the task of generating a textual narrative for a given log. Examples of system input and output are presented, and their relative merits are discussed. The final section discusses future lines of work that may be worth exploring.

Keywords: Storytelling, emergent creativity, social multiagent systems, natural language generation.

1 Introduction

Storytelling is an intellectual activity that is crucial to understand the way humans perceive the world, understand it, and communicate with one another concerning their own experience of it. As for other areas of cognition closely related with the human language faculty, research in this area has drawn interest from a very early stage, but actual progress has long been delayed by the inherent difficulty and complexity of the phenomena that require modelling. The general circumstances have not changed significantly, in the sense that adequate modelling of the human storytelling capacity is still a far off research goal. However, recent advances in multiagent systems and natural language generation have provided a set of tools that, properly integrated, can be used to build a simple model of the various subtasks involved in elementary storytelling.

An important obstacle on the road to achieving useful models of human storytelling from a computational point of view has been the lack of interaction between researchers addressing this problem from the different fields of artificial intelligence and literary studies. Recent joint efforts have started to establish a set of common assumptions, starting with the identification of a certain consensus on basic terminology and the identification of the set of subtasks involved in the broad general activity that is usually referred to as storytelling. The work of Gervás et al. (2006) distinguishes between several process that are involved in the generic process of building a story from scratch: creating a world, creating a story, and telling a story. In the past - as discussed by Callaway and Lester (2001) -, very little effort has been devoted to model computationally the task of creating a world, most efforts under the label of storytelling were concerned with creating a story, and only in the recent past has the task of actually telling a story as text been addressed.

This paper presents a decomposition of the process of creative storytelling into subtasks that are relevant for studying where and how creativity plays a role from a computational point of view. Five basic subtasks are identified: building a world to act as setting for the story (including characters, locations and possible actions), generating a set of events that take place in that world, selecting from that set of events those that are worth telling, identifying a particular sequence in which to tell them, and finding appropriate linguistic realizations for each event in that sequence. To test the model, an initial prototype is presented that operates on logs generated artificially by a social simulation built by a multiagent system. This simulation carries out the task of specifying the initial world (configuration of the simulation), and provides a log of events for a large set of characters emulating real life behaviour over a certain period of time (generating a set of events). A second module addresses the task of generating a textual narrative for a given log. This module carries out content determination (filtering the non-relevant events out of the total log), discourse planning (organizing a possibly large set of parallel threads of events into a linear narrative discourse), and sentence planning and realization (for the time being performed in a crude manner to allow readable presentation of the generated material). Of course, it is possible to create a different framework for this purpose, considering different stages. In the next sections we explain how the proposed framework is instantiated.

2 Previous Work

In order to develop this system, we have resorted to previous work in the fields of natural language generation and social simulations using multi-agent systems. A brief outline of the relevant studies is given in this section.

2.1 Automatic Story Telling

With a single exception (mentioned below in section 2.2), the label of *storytelling systems* has been used in the past mostly to refer to programs capable of creating a story, in the sense described above. No effort is made to create the world in which the story is to take place, and very rigid methods are employed to render the story in a textual form. In terms of how they model the creative process, Bailey (1999) distinguishes between three different approaches to automated story generation: *author models* - where an attempt is made to model the way a human author goes about the task of creating a story -, *story models* - based on an abstract representation of the story as a structural (or linguistic artefact) -, and *world models* - where generating a story is seen as constructing a world governed by realistic rules and peopled with characters with individual goals, and the story arises from recording how the characters go about achieving their goals.

MINSTREL (Turner, 1994) and MEXICA (Pérez y Pérez and Sharples, 2001) would be classed as examples of author models. Systems based on story grammars (Rumelhart, 1975) fall under the category of story models. Tale-Spin (Meehan, 1977), the classic Story Generator inspired on Aesop's fables, and recent efforts of story telling based on planning (Riedl and Young, 2006) are based on a world model.

A possible explanation of this diversity of approaches can be found if one considers them as partial approximations to the overall complexity of the problem. Under this interpretation, each approach is focusing on a part of the problem - the decisions required from the author, the form of the story, the restrictions imposed by the world -, and simplyfing the whole by omitting the others. Two interesting considerations arise from this hypothesis. ¿From the point of view of modelling the process, it seems that a model that takes several of these factors into account may provide greater representational power, though it may run the risk of becoming too complex to be computationally useful. ¿From the point of view of creativity, it raises the question of whether the perceived creativity of a program based on a partial model - which is modelling and controlling only a subset of the elements in play - arises from the freedom allowed in the elements that are not being modelled. Evaluation of program results should attempt to establish whether the elements that produce the impression of creativity are indeed being modelled by the program.

2.2 Natural Language Generation

Natural language generation is important for a study of storytelling because it involves both a model of the task that need to be carried out to generate a valid text - therefore partially modelling the activity of an author - and a model of the story as linguistic artefact - a story model.

Reiter and Dale (2000) define the general process of text generation as taking place in several stages, during which the conceptual input is progressively refined by adding information that will shape the final text. During the initial stages the concepts and messages that will appear in the final content are decided and organised into a specific order and structure (*content planning*), and particular ways of describing each concept where it appears in the discourse plan are selected (*referring expression generation*). This results in a version of the discourse plan where the contents, the structure of the discourse, and the level of detail of each concept are already fixed. The *lexicalization* stage that follows decides which specific words and phrases should be chosen to express the domain concepts and relations which appear in the messages. A final stage of *surface realization* assembles all the relevant pieces into linguistically and typographically correct text. The tasks of referring expression generation and lexicalization are known as *sentence planning*.

The natural language generation work presented in this paper is mainly centered around content planning.

The work of Callaway and Lester (2002) stands out as the most significant effort in the field of natural language generation to address the specific challenges posed by narrative texts. It relies on having an external planner that defines the outline of the intended story, and it carries out elaborated sentence planning to produce input for a surface realizer.

2.3 Social systems

The role of social systems in the current research is to provide a first approximation of a model of the world, which has been identified as an important ingredient of the storytelling capability. A multi-agent system (MAS) consists of a set of autonomous software entities (the agents) that interact among them and with their environment taking decitions. The agent paradigm assimilates quite well to the individual in a social system. With this perspective, agent-based simulation tools have been developed in recent years to explore the complexity of social dynamics.

The MAS presented in this paper is based on a previous social simulation by Pavon et al. (2006). In that work, the agents were developed with several main attributes: from simple ones such as sex or age, to complex ones, like for example ideology or educational level. The population in the agents' society also experiments demographic changes: individuals are subject to some life-cycle patterns. For example, they get married, reproduce and die, going through several stages where they follow some intentional and behavioural patterns. The agents/individuals can build and be part of relational groups with other agents: they can communicate with other close agents, leading to friendship relationships determined by the rate of similarity. Or, on the other hand, they can build family nuclei as children are born close to their parents.

3 Story Generation

Each of the stages outlined earlier is described in more detail.

3.1 The Social Simulation: Creating the World and the Story

Based on the ideas mentioned, several changes to the original MAS from Pavon et al. (2006) have to be made in the perspective of execution to be able to generate "life logs" of the individuals, which will be the basis for the texts describing the storyline. It is necessary to shift the point of view from trends data acquisition to vital biographies. We do not need numerical data, but semantic content that can be interpreted by the rules as we interpret them, because we want the story generation to be as close as possible to what humans might have done faced with similar sets of events. In this framework, we adapted the MAS for a Fantasy Medieval World, as it is described in León et al. (2007). Thus, for every single individual we have a Name and Last Name. Moreover, this Last Name is inherited: this will be useful for telling the stories of lineages, and for personal events. And every agent has a race, so they can be elves, humans, dwarfs... For each agent there is now a random possibility of dying, allowing the possibility that we can relate this early death to the betrayal of a friend, poisoning by a wife, a mysterious accident... This cause-consequence link is very simply implemented, only based in the last event happenned, but it should be improved with a "memory" of the agent that would lead its future actions (for ex. with a BDI arhitecture).

Following the cited objective of emulating life behaviours, the MAS presented here introduces context dependant relationships and life events: usual life events were not exciting enough to build a fantasy adventure. And so, an individual can have friends and enemies. Along his path, he can get married and have children, but he also can, randomly, suffer several spells (loss of memory, fireball or even change of sex!), kill horrible monsters (ogres, dragons), get lost in mazes or dark forests, find treasures and magic objects in dangerous dungeons,... In this way we can build a really amazing (and sometimes weird) story, with several characters that evolve and interact among them.

At the end of simulation, this collection of events, together with the agents' characteristics, is exported to a context-independent XML file. This file will be imported by the content planning module that will continue with the process of generating a story from the lives of the most interesting of these agents.

Here we present an example of the output of the social simulation with the important information of each agent. The data for every agent is listed: the initial ones, together with the next generations that appear during the simulation. Here we explain briefly the one corresponding to the individual that will be selected as star of our example story. The data for each character is divided in two main sections. The first one (Table 1) corresponds to the characteristics of the agent. Each attribute of the character has two parameters, expressed as attributes: its ID (identifier of the attribute) and its Value. The value of these keys is, of course, context-dependent: they represent aspects like its race or how religious the character is.

Id	Name	Last name	Race	Sex
i212	Jeanine	Avery	human	female

Table 1: Attributes of a character

The second main section (Table 2) is the collection of life events, associated with the time in which they took place. As in the previous sections, attributes are context-free, but values of these attributes depend on the context. Thus, we can read in the full log that in the year 515, the human Jeanine Avery suffered a spell that transformed her into a frog. Or, analyzing the chain of events, we can see that the impossible love of her youth was, after she grew to be an adult, her formal couple, giving her many children and living happily... at least for some years.

Id	Time	Action	Param
e9	515	spelled	frog
e10	515	impossible love	i229
...			
e14	526	couple	i229
e15	526	child	i258

Table 2: Events of a character

3.2 A New Rule-Based Story Planning System

The work presented in this paper is a new version of a previous content planning story generation system described in León et al. (2007). The input for this program is a description of a set of characters with their facts and attributes, and the output is a filtered and ordered set of those elements, in such a way that the final generated discourse is intended to be much more legible for a human than the original set of facts. The design is oriented to generate stories for a group of agents coexisting and creating relations between them, in this way emulating the story of a real society, with possible emerging sub-stories that can be narrated.

The previous version of the content planning system executed a ruled based algorithm to describe the story of one of the agents, telling about some important facts of some of the most important agents or characters which had any relation with the protagonist. Although the results were somehow interesting, we have carried out a new version able to generate more complex stories, not only focusing on the main character, but also on other agents. With this approach, new narrative structures can be generated, like simple conversations and changes of narrative focus.

3.2.1 Interest

The content planning system uses a rule based algorithm. The rules that directed content determination in the first version of this program were based on some numerical factors that added information to the characters, making it easier to choose which of them should be the protagonist. This new version, however, reduces the role of these values, and only computes and uses one of them, the *interest*. The rest of the information needed for creating an ordered an filtered story is included in the algorithm using more powerful rules and, as we explain later, a *focus*.

The *interest* ($I(X)$) is the importance that we assign to a character because of their facts. We have a table that gives more or less interest to each of the possible facts. Of course, these values are fully domain dependent. Formula 1 shows how to compute this factor:

$$I(X) = \sum_{i=1}^{n} f_i \cdot h\,(X, i) \qquad (1)$$

where f_i is the interest that we assign by hand for the fact i, X is the character, and $h(X, i)$ is the weight for the appearance of i in the life of X. The value of h is calculated with the type of i (what kind of fact it is) and with the attributes of X (if it is an elf, or an orc). There are, of course, some other ways of assigning this interest. However, doing it in this way we can easily put many information of the domain about the meaning and the importance of each fact.

3.2.2 *Focus on the Characters*

Interest is not enough for creating a good story. We also add some other rules to perform discourse planning, based on templates for creating an ordered story and telling stories about several characters. The rules we have chosen model in a very simple way the mental rules humans use to apply on these kind of creative tasks. However, as explained before, the previous version was only able to generate a story for a single character. We have now a different set of rules for the system: the *focus*.

The *focus* establishes which character is the main one at some stage of the story. We can consider it as a light that illuminates a particular actor in a scene. When some character has the focus, the rules generate parts of its story, as if the attention of the public centred only on a particular actor in a play. The focus can be also be understood as a "token" that is passed between the characters, and the character that possesses this "token" can add some of their facts, in the order that it decides based on some rules.

One of the most important aspects of the focus is when to change it. If the focus is only established on a single character, the system will never be able to generate a story of more than one of them. So we have to add some rules that decide when to change the focus, and which character should be the next "main actor". These rules are introduced in the next section.

3.2.3 *Rules*

The rules that govern the behaviour of the system are based on the explicit information stored in the *interest*. These rules first decide which character is the main one, choosing the character with higher interest. The story of this character will direct the main thread of the narration. So once we have chosen this character, we give it the *focus*. Then the character applies some rules to decide which facts are going to be added to the structure of the story. These rules are based on the *interest* of the facts (we only tell the most important aspects of an agent life) and the time (usually, ordering facts in a time sequence). In this way we create an ordering between the facts, creating the discourse.

When the rules for narrating a part of the life of a character decide to add to the final story a fact related to another character, we change the focus to that new character. For example, if a relation with high interest (like an important enemy) appears in the life of some character, we change the focus to that enemy, trying to show the most relevant facts about him. However, in the application of the rules along the generation process, we always keep the information of which character is the protagonist, to be able to change the focus back to him/her.

3.3 Sentence Planning

The final generation of the story is not only a nice way of showing the results. It can make the discourse interesting or boring, even if the order of the facts resulting from *discourse planning* is good or bad, respectively. Thus, we cannot ignore this step if we want to evaluate the generated content. It is not the same to say "Elrond was an Elf. He had a daughter called Arwen. Elrond was friend of Aragorn.", as to say "Elrond the Elf, father of Arwen, was friend of Aragorn the King". The final form of sentences not only gives beauty to the text, but may also convey information not actually present in the data structure. We can infer, in the second sentence, that Elrond is somebody important, as Arwen, and Aragorn is going to play a main role in the story. This knowledge is not contained in the first sentence. To achieve computational modelling of these characteristics is currently beyond the scope of this paper, but we intend to address it in future work.

Two different stages can be clearly differentiated during sentence planning: referring expression generation and lexicalization. In the work presented in this paper both of them have been treated in a simple way, with the intention of providing a first approximation to solve the problem while identifying the kind of decisions that must be taken in the future.

For the generation of references we have implemented a solution where all occurrences of the concepts mentioned are treated as definite references, and without using pronouns. Even when these references are quite simple, it is necessary to handle information about their number - singular or plural - and whether they are proper nouns or not. This information is stored in an elementary knowledge base containing the required information for each of the concepts appearing in the discourse generated by the content planning stage.

During the lexicalization stage two distinct tasks are addressed. On the one hand, for each reference appearing in the text a lexical tag must be chosen. For the moment the bare name of the concept is used, but in the future the system would work with a dictionary where every concept has assigned one or more lexical tags that are appropriate to express the meaning of the reference. On the other hand, for each action that is present in the discourse not only the lexical tag corresponding to the verb is required, but also the structure of the resulting sentence. For example, a suitable sentence for the conceptual representation of the action 'to be born' would be in passive voice and usually accompanied by a locative or temporal adjunct, as in "She was born in Rivendel" or "She was born in 1980". This information is stored in a syntactic knowledge base, where each kind of sentence is associated with the type of appropriate adjunct that can accompany it. In the current implementation the set of possible verbs is quite reduced, as well as the adjuncts corresponding to them.

3.4 An Example

Now, we show a real example of our application. The multi-agent system is capable of running parametrized simulations, changing the number of characters, probabilities of the facts, years of simulation, and all other attributes of the system. Once executed, the system generates logs in XML.

At this stage, the story generation application reads the resulting XML file, and outputs a text. This example is the result of a simulation of the life of 200 initial characters and their descendants over a time span of 80 years. The system has inferred who is the most important character, and it produces the following rendition of her mortal life:

> *Jeanine Avery was born in 520. Jeanine Avery was saved by the priest. Jeanine Avery killed the ogre. Jeanine Avery was involved in the battle. Jeanine Avery was enchanted with the marvellous spell of the frog. Jeanine Avery killed the dragon. Jeanine Avery was lost in the forest. Jeanine Avery met Luisa Brandagamba. Luisa Brandagamba was born in 529. Luisa Brandagamba met Jeanine Avery. Jeanine Avery killed the ogre. Jeanine Avery fell desperately in love with Bobbie Beasttongue. Jeanine Avery inherited the castle. Jeanine Avery met Pogor Brandagamba. Pogor Brandagamba was born in 529. Pogor Brandagamba killed the ogre. Pogor Brandagamba met Jeanine Avery. Jeanine Avery grew up. Jeanine Avery fell desperately in love with Bobbie Beasttongue. Jeanine Avery met Haurk Avery. Haurk Avery was born in 542. Haurk Avery found the magic sword. Haurk Avery met Jeanine Avery. Jeanine Avery was lost in the labyrinth. Jeanine Avery found the treasure. Jeanine Avery was enchanted with the marvellous spell of the memory. Jeanine Avery was enchanted with the marvellous spell of the frog. Jeanine Avery was involved in the battle. Jeanine Avery killed the ogre. Jeanine Avery killed the dragon. Jeanine Avery was involved in the battle. Jeanine Avery was lost in the forest. Jeanine Avery found the treasure. Jeanine Avery killed the dragon. Jeanine Avery was betrayed and killed by Morrain Avery.*

4 Discussion

An important advantage of the proposed model is that it allows separate discussion of the degree of creativity achieved at each stage, and considerations of whether the corresponding creativity can be attributed to the program, the programmer, or randomness.

4.1 World Construction: How Creative is it?

Even though we have defined a full context where the action takes place, there still remains lot of work to do for building the world. This MAS can be configured in many ways that lead to completely different results. But this fact does not mean that there is a way of predicting the result: as a social system, its nature is non-deterministic. Although we can partly control the agents' behaviour, for the most part it remains uncontrolled. Only through evaluation and statistics can wededuce which configuration is the most convenient to our aim.

To achieve this, the system was tested changing the values of the parameters and analyzing the results obtained for each configuration. Initial testing was directed to find the base parameters for other testing processes. It revealed that hundreds of individuals were needed to provide enough interactions to extract emergent behaviour and to express enough complexity, following logical reasoning. We found too that if we wanted to tell the life story of a person, we needed to know at least a number of decades of its life events. On the other hand, we can not use thousands of agents due to efficiency issues (the analysis of a log of hundred of MB is costly in computational terms). Thus, we left constant the size of the initial population (200) and the time of simulation (50 years) for all the configurations tested (together with other minor parameters like the space size or those dealing with graphical presentation), and only the demographical parameters are modified. We chose 200 agents because it is a number that the system can handle in reasonable periods of time, but big enough to produce good results.

We say a system is "stable" if it returns a similar output in different executions with the same configuration. New evaluations with those fixed parameters revealed that the stability of the social simulation system was critically affected by the parameters chosen. Furthermore, we found that the same parameters were dramatically influencing the diversity of the population. And diversity is the main path to find creativity.

So, we identified these critical parameters as:

- Mean of children per couple: a simple number that specifies the mean of the normal distribution that define how many children a couple have. It has two typical parameters, corresponding to the means in a developed country (2) and in a typical African one (5).
- With/without initial kids: a complex parameter that reflects the amount of kids in the initial population. In the original sociological MAS there were no kids initially, because all the data of the agents were imported from surveys... and kids do not complete surveys. We can force the appearance of kids reducing

the ages of all the agents of the initial population.

The importance of both parameters can be easily explained. About the first one, we can say that if every couple has more children, there will be more interactions, more friends, more complexity. If we add the fact that sons and daughters are born close to their parents forming family nuclei, the system dynamics tend to self-organize in clusters of friends and families concentrated in the space. Newly born agents grow in a rich environment full of people, so they can generate lots of events and easily find a spouse, better than more isolated ones. Evolution does the rest. After decades of simulation, only the clusters survive and grow.

On the subject of whether to have children in the initial population or not we can say that, without them, the size of the population tends to get lower and lower with time, because there is not a new generation that substitutes the oldest ones that are dying. And moreover: the characters with an "interesting life" are the ones that have enough time to do lot of "interesting things"... If lot of them begin the simulation with 45 years or more, and they tend to die around 65, they do not have much time to "be heroic". The same could be said if an agent is born 5 years before the end of simulation: its probabilities to be "interesting" are very rare.

We can define four different configurations based on these two parameters: Without/2, Without/5, With/2, With/5. We will analyze some evaluation results of them as shown in the following figures and explanations. These figures reflect the analysis of dozens of executions and tests.

In the four configurations, we will observe the total number of individuals simulated (the 200 initial ones plus the born ones) and the size of the family (average and maximum, minimum gives no information). The initial people have no parents and no initial relationships between them.

In Without/2 the number of individuals has a very high stability, with variations around the 5% between executions. Because to both critical parameters tend to reduce the population there are only a few births. Thus, average size of family is very close to the minimum. In Without/5, with each parameter pushing in a different direction, we can see that the "Without" one is stronger: the population still decreases. The family size doubles and the maximums are incredibly higher: three times the Without/2 (around 15).

In With/2, again we can see how "With" prevails: the population increases, but only around a 45%. Here we appreciate a logical increase of the average family size (around 3.0), but less than it could be expected: although here we have a rise of population, it is because we avoid deaths, not because children. We can realize the difference better looking at maximum family size, that here reaches only 10. In With/5, we have what we could expect: an incredible grow of the population size. With crazy executions always completely different (with differences that reach the 200 births), it results an average growth of 275%, an average family of 6.2 and a maximum family size of 18. Now, none of these facts could surprise us. The unstable executions of With/5 are shown in Figure 1 compared with the stability of others.

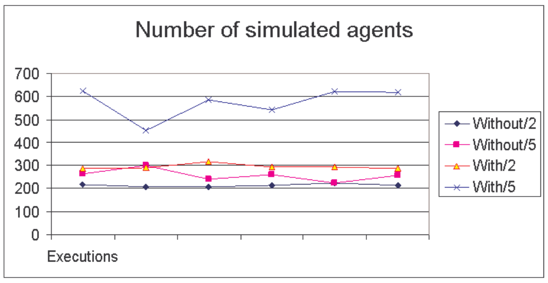


Figure 1: Comparison between configurations, attending to the number of simulated agents

After reviewing all the possibilities, the chosen one would be the one that has a good amount of interactions together with lot of diversity in the population and a reasonable family size.

Dealing with races, we can see how the micro-decisions (taken by the agent) determine the total evolution in a kind of butterfly-effect (chaotic). We force the initial population to have approximately 20% of each one of the five races. But it does not matter which configuration we choose: the percentage will never remain stable. The reason is simple: each agent becomes friend or enemy of someone depending on the similarity between them. And when an agent has to choose its spouse, it chooses the most similar of its friends (some other minor restrictions are included). But an elf and an orc have nothing in common: they dislike each other, and will be extremely difficult for them to marry. Because of that, the number of orcs tends to decrease (if one is in a environment without orcs, he will not find a couple nor have children). Humans can easily cross with other races (according with fantasy middle-age stories), so their adaptability allows them to increase their number more than other races. This way we are modelling the fantastic world, and it makes sense: we will not find strange when the human Aragorn hates orcs and gets married with the elf Arwen. Figure 2 reflects this explanation graphically.

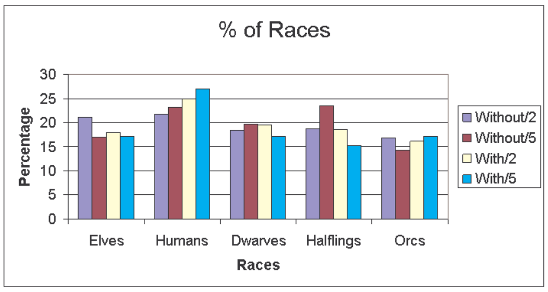


Figure 2: Percentage of races depending on the configuration

Even though we have reviewed the main possibilities of the demographical model types, we cannot decide which one would be the best one for our tale. This is a task of the next step: story building.

4.2 Story Construction: How Creative is it?

We have a complete world, with a demographical evolution, that serves as context. Now we focus on the facts to be told in the story, that have two main characteristics: they have to be "interesting" (our characters are adventurers, not trees that only know how to grow, reproduce and die) and they have to be restricted to this world (Aragorn will not have a helicopter). Because of the first characteristic, we chose to include the random context events (as it has been described) and the relationship of "enemies". Because of the 2nd one, we chose the events carefully, applying our knowledge of the fantasy context. In this way we can analyze, as was done in the previous subsection, the friends, enemies and number of events of each configuration, from the "interest" point of view. Maximums are particularly interesting because they will reflect the "heroic" characters, that will do more things than usual, and so will be more attractive for an exciting story.

For Without/2, with not much population, the interactions between agents are not significant. This simulation is dramatically poor: an average of not even 4 friends per individual and maximums of 10 reveals it. With many dying in the beginning, the average number of events does not grow more than 20... Although this amount is doubled by the "heroes", it is too poor to be considered. We should have much more "interesting" ones to analyze, so we can decide about a really amazing one, or about crossing two interesting and connected stories. For Without/5 we increased the amount of friends/enemies by a 50%, but still not enough: 5 friends and 2 enemies for a whole life is not what we are expecting. The events have nearly no increase, and the maximum is just a 15% higher.

In With/2 we can see, at last, good averages. The amounts of friends/enemies have doubled respect to Without/5, and in the special characters we can see dozens of friends/enemies. Besides, a big increase in the number of events is observed . This configuration could be selected for our purposes. With/5, as in the other subsection, gives crazy results. We can see a huge increase in the number of friends/enemies (the maximum, with 62, doubles With/2!) and, even though the average of events did not change a lot, the maximum is incredibly high: an average maximum of 125 means that the amount of heroic characters is very significant. Figure 3 shows the main event results.

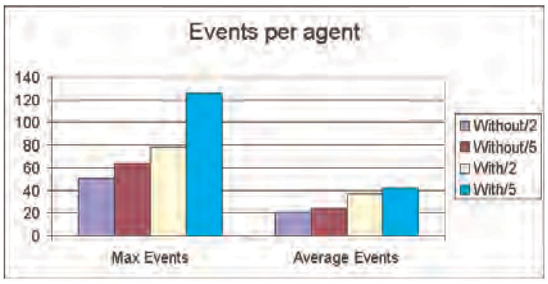


Figure 3: Comparison between configurations, attending to the number of events per agent

A note about efficiency: even though all the configurations ran with no problems of time or memory, the With/5, due to its huge amount of agents and interactions, is significantly slower than the others. Moreover, it outputs a XML log much bigger than the other configurations. Attending to the whole discussion and reviewing the possibilities commented, the chosen configuration will have a good amount of interactions, together with the diversity necessary for finding many "heroes". The With/2 and With/5 are good candidates. Choosing between them depends only on the length and complexity of the story that wants to be told and efficiency issues of the other modules.

A good story must explain why a fact occurs in the story. Only with random events this is, of course, not possible (there is no explaining possible for a random event). We can see, in the example (3.4), that in the story there are some gaps. These gaps are relative to the randomness of the system. This is an important issue that has to be improved.

4.3 Content Planning: How Creative is it?

To obtain a creative content planning system is definitely not an easy task. A truly creative program able to emulate human creativity should have mechanisms for understanding the source logs, creating an intention for the discourse plan, and performing some operations for threading the final story. Of course, nowadays this a very ambitious project, and we have to give little steps towards this ideal system.

In this paper we have presented some progress. As explained before, textual content generation from the facts of the story are generated with rules. This set of rules can be more or less large and complex, but in this kind of systems they will always direct the generation, and thus the quality and the creativity of the resulting text.

In this sense, the creativity relies on the quality of the rules, which is directly linked to the creativity of the human responsible for writing them. In this way, we can say that the human puts the creativity into the system, and the system only reproduces the information written by the human.

This information is not only explicitly deposited on the system with the rules, but also with the *interest* and the *focus*. As explained before, *interest* is stored in a table created by a human, and the focus traffic is guided by rules. Again, the creativity of the content planner is dependent on the human creativity. It is important to note that in this version the rules that govern focus changing are not very good, and the focus swings too much from one character to another. This has to be improved.

4.4 Sentence Planning: How Creative is it?

For the moment the implementation of the sentence planning stages of the process are systematic, and therefore they can not be considered creative. Certain improvements are possible which may result in a higher quality of the output texts. However, this improvement in quality is in truth more related with issues of style than with creativity.

During the generation of referring expressions some of the decision processes involved can be improved. For instance, imagine we are referring to a girl about which the system knows that is pretty and is daughter of a king. If the references to this concept are directly translated from the available information we would obtain sentences such as “The girl was pretty. She was the daughter of a king”. However, the final text would be more natural if it is capable of inferring that the girl is a princess and referring to her as a princess in the rest of the text. This involves providing the system with a certain amount of knowledge that it can use to simulate more inteligent behaviour.

Also in the lexicalization tasks there is space for improvement. When choosing the lexical tag to use for a concept, it seems common that the dictionary had more than one word for each of these concepts. Depending on the previous appearances of the concept, the system can choose between synonyms or hypernyms using heuristics about the style of the discourse in a specific moment, or the emotion that the text is trying to transmit.

5 Conclusions

We have presented a system where interactions between agents over a long period of time can be told in natural language automatically.

We have shown a particular way of generating the stories, based on rules. We have explained a three-step process for performing this task, and we have verified that for *discourse planning*, the rule-system is very dependent on the domain, and the desired type of story.

The results of the system are less impressive - when rendered in a readable text format - than they might have been if the system included an elaborate sentence planning module. The current version is just a skeleton implementation that lets down an otherwise acceptably selected and planned discourse.

The division of story telling into five tasks is envisaged as a generic analysis of the process, in the sense that it should be applicable to all storytelling systems. This does not apply to the sequential manner in which they are carried out in the prototype described. For different systems, the results of some of these tasks may be provided as input (for instance, descriptions of the world are given to story telling systems based on planning, or discourse plans provided to Callaway’s StoryBook system). A different alternative is to avoid altogether explicit modeling of some of these intermediate results. For instance, the creation of the story world may not take place explicitly within the system, and simply be left to emerge in the reader’s mind from the sequence of events that is built by the system. This does not make the proposed model less valid. Whether a particular task is modeled explicitly in any given system, outsourced to the user (or a different system), or left to emerge implicitly from the results of other tasks, it remains true that a story (and hence a story telling system) may/should be evaluated at these five different levels.

Acknowledgements

This work is partially supported by the Spanish Ministry of Education and Science project TIN2006-14433-C02-01, and research group grant UCM-CAM-910494, jointly funded by Universidad Complutense de Madrid and the Comunidad Autónoma de Madrid (Dirección General de Universidades e Investigación).

References

Bailey, P. (1999). Searching for storiness: Story-generation from a reader’s perspective. In *Working Notes of the Narrative Intelligence Symposium, AAAI Fall Symposium Series.*, Menlo Park, CA. AAAI Press.

Callaway, C. and Lester, J. (2001). Narrative prose generation. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 1241–1248, Seattle, WA.

Callaway, C. B. and Lester, J. C. (2002). Narrative prose generation. *Artif. Intell.*, 139(2):213–252.

Gervás, P., Lönneker-Rodman, B., Meister, J. C., and Peinado, F. (2006). Narrative models: Narratology meets artificial intelligence. In Basili, R. and Lenci, A., editors, *International Conference on Language Resources and Evaluation. Satellite Workshop: Toward Computational Models of Literary Analysis*, pages 44–51, Genova, Italy.

León, C., Hassan, S., and Gervas, P. (2007). From the event log of a social simulation to narrative discourse: Content planning in story generation. In *AISB’07, Artificial and Ambient Intelligence*.

Meehan, J. R. (1977). Tale-spin, an interactive program that writes stories. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, Cambridge, Mass. Morgan Kaufmann.

Pavon, J., Arroyo, M., Hassan, S., and Sansores, C. (2006). Simulacion de sistemas sociales con agentes software. In *Actas del Campus Multidisciplinar en Percepcion e Inteligencia, CMPI-2006*, volume I, pages 389–400.

Pérez y Pérez, R. and Sharples, M. (2001). Mexica: a computer model of a cognitive account of creative writing. *Journal of Experimental and Theoretical Artificial Intelligence*, 13(2).

Reiter, E. and Dale, R. (2000). *Building Natural Language Generation Systems*. Cambridge University Press.

Riedl, M. and Young, R. M. (2006). Story planning as exploratory creativity. *New Generation Computing - Special Issue on Computational Creativity*, 24(3-4).

Rumelhart, D. (1975). Notes on a schema for stories. In Bobrow, D. G., editor, *Representation and Understanding. Studies in Cognitive Science*, New York. Academic Press.

Turner, S. R. (1994). *The Creative Process: A Computer Model of Storytelling*. Lawrence Erlbaum, Hillsdale, NJ.

On the Fly Collaborative Story-Telling: Revising Contributions to Match a Shared Partial Story Line

Pablo Gervás
Universidad Complutense de Madrid
28040 Madrid, Spain
pgervas@sip.ucm.es

Rafael Pérez y Pérez **Ricardo Sosa** **Christian Lemaitre**
Universidad Autónoma Metropolitana, (Cuajimalpa)
México D.F., México
rperez@correo.cua.uam.mx
rsosa@correo.cua.uam.mx
clemaitre@correo.cua.uam.mx

Abstract

Computational improvisation is a challenging topic. It involves collaborative creativity, the modelling of interesting cognitive process like those needed to keep the coherence and interestingness of an emergent story, the ability to foresee possible interesting directions that the improvisation might take, etc. In this paper we present an architecture for story-telling improvisation. It is based on the engagement-reflection computer model for plot generation. It involves the interplay of two agents in order to generate a novel, coherent and interesting story. Our purpose is to provide an analysis of the key requirements to develop a computational improviser and the solutions we envisage to achieve this goal.

Keywords: Colaborative storytelling, improvisation, engagement-reflection.

1 Introduction

Improvisation can be defined as the act of creation of a work or its performance in real time, individually or by a sequence of contributions by a number of interacting agents. Improvisation is known to be a type of collaborative creative activity that takes place extemporaneously with continuously updating preparation but without prior planning. In improvisation, a combination of planned and unplanned actions takes place. The contributions of each improvising actor often follow three basic restrictions: they must be consistent with the contributions of other players, they are expected to result in an interesting plot that emerges from their interaction with the rest of the players, and they must be produced by avoiding noticeable gaps in the run of the scene.

From the viewpoint of computational creativity, this set up presents a number of interesting questions. On one hand, the restriction on the quality of the emergent material suggests that some kind of shared intentionality may be required to drive the production of each actor's contribution. Actors may be searching for particular effects when they produce certain contributions. To a certain extent this may be modeled as some kind of preparation[1] activity, during which the actor contemplates possible effects of his immediate actions and produces his contribution based on that preparation.

On the other hand, the restriction on overall consistency of the set of contributions implies that actors must continuously consider the contributions of other players. A contribution prepared by one player may need to be revised, altered or even scratched altogether if another player generates conflicting material before that contribution is actually executed. The fact that interaction takes place in sequences without gaps complicates matters further, since it precludes the elementary solution of waiting until everybody else's contribution has finished before starting to prepare one's own.

This problem is worth studying both from the point of view of understanding how humans address it and from the point of view of devising computational methods for emulating this behaviour in particular situations. However, theatrical improvisation involves too many complex levels of interaction to be modelled successfully in computational terms: a text must emerge from the interaction, but other ingredients such as diction, gesture, body language... play too crucial role to be dispensed with without compromising the validity of the analysis.

A possible solution is to try to find a simpler problem that retains the fundamental issues concerning preparation, revision, emergent quality and real-time interaction, but has a lower complexity of the material to be considered. In this paper we put forward a model for on-the-fly collaborative storytelling that may satisfy these criteria. Two story tellers take turns in advancing a shared story line. While one contributes, the other one listens. Because he will be expected to take over as soon as the speaker stops, he cannot postpone the task of preparation until the speaker has finished. So he prepares ahead a ten-

[1]Preparation defined as "to be prepared: to be in a state of readiness, ready; to be mentally ready, inclined, disposed; to be in a condition or position to do something" (Oxford English Dictionary, www.oed.com) is used in a more flexible way than the more definite nature of planning used in the traditional AI literature.

tative sketch of his own contribution. Any conflicts arising with the other story teller's contribution as it emerges will force him to revise his prepared contribution.

We believe this model to contain all the elementary details that puzzle us in theatrical improvisation, while being restricted to the somewhat simpler task of story generation, for which a number of computational solutions already exist. In this paper we propose an architecture for engaging two such existing computational programs for story telling - in truth, two copies of the same solution, possibly running under different configurations - in an exercise of on-the-fly collaborative storytelling. We discuss the restrictions that the specification imposes on the story tellers, and we consider how their collaborative creative tasks can be modelled.

Improvisation has been part of theatre performance for centuries; it is well documented that performers of the *Commedia dell'arte* were excellent improvisers[2]. During the first half of the twentieth century in Chicago, Viola Spolin introduced the theatre games based on improvisation; since then, the number of improvisation practices has increased. We are interested in furthering our understanding of the relation between improvisation and creativity within computational settings. Some improvisation issues that we identify as key elements for this research include:

- Criteria must be established to evaluate the quality of improvisations and the means to validate this measure.
- Coherence must be maintained during an improvisation. This is an important challenge since actors cannot modify what has already been told.
- Actors have different knowledge and experiences and therefore different ways of interpreting the world.

2 Previous Work

The work described in this paper involves story telling, improvisation, and interaction between programs that can roughly be classed as agents in the sense that they communicate and exchange data in a colaborative effort to produce a common result. For each of these elements, a brief summary of the relevant background is provided in this section.

2.1 Automatic Story Telling

Of the various approaches to automated story telling described by Bailey (1999), those based on modelling the processes that a human author follows in generating a story are most interesting from the point of view of modeling creative endeavour from a computational point of view. The work of Turner (1994) on the MINSTREL system was pioneering in the sense that earlier attempts at automated storytelling - such as the work of Meehan (1977) or Rumelhart (1975) - focused more on modeling the world about which stories are told or the actual form of the story as a linguistic artefact, respectively. Although several research efforts have addressed storytelling in different ways since then, it is not until the work of Pérez y Pérez and Sharples (2001) on the MEXICA system that modelling the actual processes of creative story composition has been addressed especifically.

The work presented in this paper presents an extension of a similar analysis to the modifications to the creative process induced by a colaborative improvisational setting.

2.2 Improvisation as Collaborative Story-Telling

The on-line Webster dictionary defines improvisation as "an unplanned expedient" or "a performance given without planning or preparation". It is clear that improvisation is something different from planning. However, Moraes and da Rocha Costa (2002) claim that planning can be understood as improvisation under external constraints. In their model, there is a "director" who is responsible for providing the actors with a full script of the story to represent; so, the actors' job consists in finding ways of reaching the goals imposed by the script. This has been, in fact, a research line of hierarchical planning where planning and acting can be interleaved. By contrast, we consider improvisation a collective activity where the plot (or script) emerges as result of the interaction between agents.

Philip Agre - in Agre and Chapman (1987); Agre (1997) - has characterized computational improvisation as the continual dependence of an agent's action upon its circumstances. This proposal interprets improvisation as a running argument that an agent continually updates among various alternatives. These options together form a dynamic argument structure which undergoes constant change as a result of agent activity and its impact on the world, including other agents. Under this approach, the emerging behavior is an considered as epiphenomenon of the interactions between agents and their world. There are a few basic constraints mentioned in the literature of theatre performance that a good improvisation must fulfill. Following Trastoy (2005), some of these constraints are listed below.

Improvisation is a sort of story-telling and the whole performance must have some basic structure such as introduction to the problem, development and resolution.

Nothing should be agreed in advance within the group of actors. The public might suggest a topic to be developed during improvisation and actors can take some seconds to define basic issues about roles and other matters, but nothing about the plan of the story.

The dynamic of the story is driven by conflicts. The story is in fact a collective search for solutions to those conflicts. During the unravelling of the plot new conflicts might arise. The challenge for actors is to keep the coherence and interestingness of a dynamic and unpredictable story, finishing with a good synthesis of the different problems.

The role of the director of an improvisation troupe is quite different from the role of the director in traditional theatre. The former is responsible for the general setting of the performance while the latter controls every aspect of the play.

Thus, an improvisation performance is a sort of collec-

[2]This article was originally published in Bellinger (1927)

tive story-telling game where the golden rule is that no actor must block the story initiated by its predecessor. That is, he can never say something like “No, what he said is not true, the truth is...” Besides that rule, actors are free to generate the next segment of the story as they want.

Our approach to computer improvisation as story-telling incorporates some of the main characteristics of the improvisation troupes. We envision a set of agents who may play specific roles in the story. The role of the director is played by the programmer who defines some basic features of the story: the length of the story, the maximum length of each “intervention” of the agents, the number of characters, which character is assigned to which agent, the agent who will start the performance, etc.

2.3 Agent Architectures

The Open Agent Architecture (OAA) Cheyer and Martin (2001) is a framework for developing multi-agent systems intended to enable more flexible interactions among a dynamic community of heterogeneous software agents. The operation of the architecture is based on the idea of delegation: agents do not hard-code their interactions (method calls or messages) in a way that fixes how and whom they will interact with, instead the interactions between OAA agents are expressed in terms of needs delegated to a Facilitator agent. This Facilitator agent coordinates the agent community so that it can achieve its task. It does this by providing services such as parallelism, failure handling, and conflict detection, which relieves each client agent from having to worry about these issues itself. OAA’s Distributed Agents are simply programs - or wrappers around programs - that share several common functionalities, and which are possibly distributed across various machines.

3 The MEXICA Story Telling System

MEXICA is a computer model of creativity in writing that develops frameworks for short stories. It is inspired by the engagement-reflection account of writing given in Sharples (1999). In MEXICA a story is defined as a sequence of actions. Each action has associated a set of preconditions and post conditions, defined by the user of the system, which are comprised by emotional links and tensions between characters. Emotional links are represented as a continuum between hate and love with discrete values ranging from -3 to +3. In this way, the precondition of the action *Hunter killed Jaguar Knight* might be that the hunter hates the knight (an emotional link of intensity -3; see first line of Table 1); the post condition of the action *Princess decorated Eagle Knight* might be that the knight is very grateful towards the princess (an emotional link of intensity +2; see second line of Table 1). In MEXICA, the tension in the story increases when a character is murdered, when the life of a character is at risk, when the health of a character is at risk (i.e. when a character is hurt or ill), or when a character is made a prisoner. Like emotional links, tensions can be employed as preconditions or post conditions. Actions also might include post conditions that deactivate tensions. In this way, the action *Princess healed Jaguar Knight* has as a precondition the fact that the knight must be injured or ill (a tension due to health at risk) and as a post condition the fact that the knight has been cured (the tension is deactivated) and that the knight is very grateful towards the princess (an emotional link of intensity +2) (see third line of Table 1). Finally, MEXICA includes inferred tensions, i.e. tensions that are activated when the system detects that: 1) two different characters are in love with a third one (tension due to love competition); 2) when a character has two opposite emotions towards other one (tension due to clashing emotions); 3) and when a character hates other character and both are located in the same position (tension due to potential danger). If the conditions that activate an inferred tension disappear, the tension is deactivated. Each active tension has associated a value that the system records each time an action is performed. In this way, the system represents as a graph the value of the tension in the tale over story-time. A story is considered interesting when it includes increments and decrements of the story-tension, e.g. if a princess is kidnapped (an increment in the tension) and then rescued (a decrement of the tension). All actions’ post conditions are recorded in a structure known as the story-context. So, the context represents the state of affairs in the story in progress. MEXICA has two core processes: the creation of knowledge structures in memory and the plot generation.

3.1 Construction of Knowledge Structures

MEXICA builds its knowledge structures from a set of narratives known as *previous stories*. Previous stories are provided by the user of the system and they are composed of sequences of actions. So, previous story 1 is formed by action 1, action 2, action 3, and so on. For the sake of a clearer explanation, we first describe how story-contexts are updated when MEXICA processes the previous stories and then we elaborate the explanation to clarify how knowledge structures are created. The process of updating story-contexts work as follows: 1) MEXICA takes the first action in the first previous story, triggers its post conditions and updates the story-context; 2) MEXICA takes the second action in the first previous story, triggers its post conditions and updates the story-context; and so on. In this way, each time an action is performed the story-context is updated. So, we can refer to the story-context after action 1 is performed as context 1, to the story-context after action 2 is performed as context 2, to the story-context after action 3 is performed as context 3, and so on. Or we can say that action 1 generates context 1, action 2 generates context 2, action 3 generates context 3, and so on. Notice that context 2 is not necessarily made up by the addition of the post conditions of actions 1 and 2. As mentioned earlier, some action’s post conditions might deactivate tensions between characters, and inferred post conditions might become active or inactive at any moment. Thus, the story-context is a very dynamic structure that progresses over story time.

Thus, the process to build knowledge structures works as follows:

1. MEXICA takes the first action in the first previous story, triggers its post conditions and updates the

Precondition	*Action*	*Postcondition*
The hunter hates the knight (an emotional link of intensity -3)	**Hunter killed Jaguar Knight**	
	Princess decorated Eagle Knight	The knight is very grateful towards the Princess (an emotional link of intensity +2)
The knight must be injured or ill (a tension due to health at risk)	**Princess healed Jaguar Knight**	The knight has been cured (and therefore the tension has been deactivated) The knight is very grateful towards the Princess (an emotional link of intensity +2)

Table 1: Three actions with their pre and post conditions (defined by the user of the system).

story context creating context 1. Then, it copies context 1 into a new structure created in memory known as atom 1. Next, it copies the following action in the previous story - in this case action 2 - into atom 1. In this way, atom 1 is linked to action 2.

2. MEXICA takes the second action in the first previous story, triggers its post conditions and generates context 2. Then, it copies context 2 into a new memory structure known as atom 2. Next, it copies action 3 into atom 2. So, atom 2 is linked to action 3.

3. If atoms 1 and 2 are alike, the system copies the action linked to atom 2 into atom 1 and destroys atom 2. So, atom 1 is linked to action 2 and action 3.

The following lines exemplifies this process. Imagine that the first previous story includes the following sequence: *Farmer wounded Jaguar Knight*; *Princess cured Jaguar Knight*; *Jaguar Knight murdered Farmer*; *The End* (see Figure 1). The first action, where the knight is wounded, generates context 1 which is comprised by the tension *Jaguar knight's life is at risk* and the emotional link *Jaguar Knight hates Farmer*. MEXICA copies context 1 into memory, creates atom 1 and links the following action in the sequence (in this case *Princess cured Jaguar Knight*) to atom 1 (see case *a* in Figure 1). Notice that, within atoms, characters are substituted by variables. In this way, atom 1 represents the knowledge that when the life of a character X is at risk (where character X is any character) and character X hates character Y (where character Y is any character but X) a logical way to continue a story is that a third character Z heals character X. This information will be essential during plot generation. Next, the system triggers the post conditions of the second action in the story, i.e. when the knight is healed. So, context 2 is created; it is comprised by the emotional link *Jaguar knight is very grateful towards the Princess* and a second emotional link *Jaguar Knight hates enemy*. Notice that the tension Jaguar knight's life is at risk is deactivated as a result of the princess curing the knight. So, it disappears from the context. Context 2 is copied into memory to create atom 2 and action 3 is linked to such an atom (see case *b* in Figure 1). MEXICA takes action 3 and triggers it post conditions; however, because this is the last action in the story the process stops. The same process is repeated for each previous story. At the end, if the system is provided with enough stories, each atom in memory might have several linked actions. Each atom in memory represents a possible state of affairs in the story world in terms of emotional links and tensions between characters. Linked actions provide different routes that a narrative can follow during story generation given a specific story-context.

3.2 Plot Generation

There are two core processes that interact during plot generation: engagement and reflection (see Figure 2). During engagement the system produces sequences of actions as follows: an initial action is selected; MEXICA triggers all its post conditions updating the story-context; the system employs the story-context as cue to probe memory and tries to match an atom that is equal or similar to it; the system retrieves all the actions linked to the matched atom and selects one at random as the next action in the story; the system updates the story-context and the engagement cycle starts again. If the system cannot match any atom in memory an impasse is declared. By default the cycle repeats until three actions are generated or an impasse is declared. Then, the system switches to reflection.

During reflection the system:

1. Verifies that the preconditions of all actions generated during engagement are satisfied (notice that preconditions are ignored during engagement). If necessary, the system inserts actions in the story produced so far to satisfy preconditions.

2. Evaluates the interestingness and novelty of the story in progress. A story is interested when it includes increments and decrements of tension (e.g. if the princess is kidnapped and then rescued); a story is novel when it is not similar to any of the previous stories (MEXICA compares sequences of actions between the story in progress and all the previous stories).

3. Breaks impasses.

Then, the system switches back to engagement and the cycle continues. The interaction between engagement and reflection generates MEXICA's output. Atoms are knowledge structures comprised by emotional links and tensions. They are general enough to enclose different alternatives to progress a story, but at the same time they are specific enough to drive in a coherent way the development of a tale. So, a narrative can be expressed in terms of clusters of emotional links and tensions between characters that progress over story time. We exploit this characteristic to propose an architecture for improvisation.

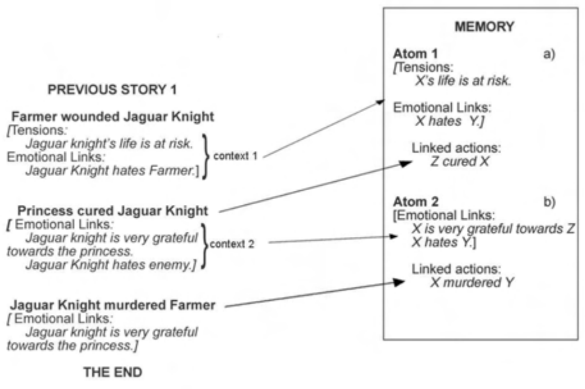

PREVIOUS STORY 1
Farmer wounded Jaguar Knight
[Tensions: Jaguar knight's life is at risk.
Emotional Links: Jaguar Knight hates Farmer.]
context 1
Princess cured Jaguar Knight
[Emotional Links: Jaguar knight is very grateful towards the princess. Jaguar Knight hates enemy.]
context 2
Jaguar Knight murdered Farmer
[Emotional Links: Jaguar knight is very grateful towards the princess.]
THE END
MEMORY
Atom 1 a)
[Tensions: X's life is at risk.
Emotional Links: X hates Y.]
Linked actions: Z cured X
Atom 2 b)
[Emotional Links: X is very grateful towards Z X hates Y.]
Linked actions: X murdered Y

Figure 1: How atoms are created in memory: a) illustrates atom 1 comprised by one emotional link and one tension and linked to the action Z cured X; b) shows atom 2 comprised by two emotional links and linked to the action X murdered Y. Z, X and Y represent variables. Atom 1 is built from context 1 and atom 2 from context 2.

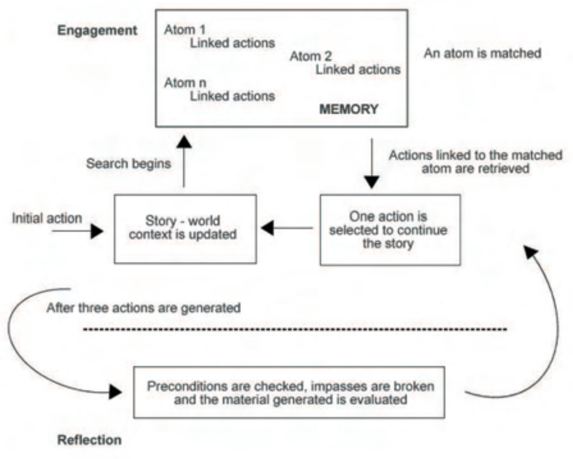

Engagement
Atom 1 Linked actions
Atom 2 Linked actions
Atom n Linked actions
MEMORY
An atom is matched
Search begins
Actions linked to the matched atom are retrieved
Initial action
Story - world context is updated
One action is selected to continue the story
After three actions are generated
Preconditions are checked, impasses are broken and the material generated is evaluated
Reflection

Figure 2: The engagement-reflection cycle

4 An Architecture for On the Fly Collaborative Storytelling

Two aspects determine how colaboration takes place between story telling programs: how each program addresses the task of creating stories in this way, and how the colaboration between the story tellers is orchestrated. We are assuming that from the point of view of creativity, the first aspect is fundamental, whereas the second aspect concerns a tecnical issue of interconnecting two systems. An ideal solution to this second problem should be independent of the actual storytelling processes employed by each participant.

4.1 The Participating Storytellers

We employ two agents: MEXICA 1 (M1) and MEXICA 2 (M2). The basic process of our system will work as follows:

- the user provides an initial action (action 1).
- M1 and M2 create their own story-context (so, we have context 1 of M1 and context 1 of M2).
- M1 generates one action to continue the story (action 2), updates its story-context (creating context 2 of M1) and communicates to M2 the action 2.
- M2 receives action 2, updates its own story-context (creating context 2 of M2) and generates a new action (action 3) to continue the story.
- M2 updates its own story-context (creating context 3 of M2) and communicates to M1 the new action (action 3) in the improvisation.
- M1 updates its story-context (creating context 3 of M1), generates a new action (action 4), and so on.

In such a setting each one agent would only start preparing its actions once the other one had finished his contribution. The improvisation problem would become equivalent to the two story-tellers taking turns in extending the story. Several modifications to the basic process can be applied to enrich the simulation. The agents involved in the basic model basically act in the role of authors, because there is no difference between the way they respond to a contribution of the other agent and the way they respond to a contribution of their own. This can be changed by extending the number of actions that each agent can contribute in his turn. If an agent can produce more than one action without passing the turn to the other agent, this forces the passive agent to model a new attitude: that of a listener of the story. At any point during an improvisation, one agent would be operating in speaker mode, and the other one in listener mode. While in speaker mode, an agent generates actions and communicates them to other participants in real-time. While in listener mode, an agent may generate actions, but it does not communicate them. Instead, it puts them in a store of tentative contributions to wait until its turn comes to operate in speaker mode. This store of contributions must be revised whenever a new contribution is received from a speaker.

An agent acting in *speaker mode* would operate as follows:

- At the start of its turn, it communicates to other participants all those of its stored tentative contributions that were not in conflict with those communicated during previous turns, and which have not already been contributed by other agents.
- Then it carries on generating new actions. At each stage of the turn, it generates one action to continue the story, updates its story-context and communicates to the other participants the new action.
- It carries on in this way until his turn finishes.
- When it does, the agent switches to listener mode and some other agent switches to speaker mode.

An agent acting in listener mode must carry on two parallel process that it must combine to result in a single context. On one hand it must silently generate tentative contributions to the ongoing storyline. On the other hand, it must keep its version of the story line updated with whatever contributions are provided by other agents acting in speaker mode. To achieve this, a listening agent must keep a store of his tentative contributions, which are possible continuations of the story but which have no fixed place in the story line until he actually communicates them to other participants. To achieve this, it must maintain two different versions of the context: the *current context* corresponding to contributions actually communicated by speaker agents, and a *tentative context* resulting from applying to the current context the list of his tentative contributions.

The combination of the two tasks of an agent in *listening mode* would operate as follows:

- While it receives no contributions from outside speakers, it generates possible actions to include in its contribution when its turn comes (but it does not communicate them to other participants!), and it updates its tentative context with them.
- When an agent in listener mode receives an action from another agent acting in speaker mode, it updates its own current story-context by adding that action, and it revises its tentative context in the following way: if the new action was already contemplated in its tentative context, it is retracted from it (and all tentative contributions beyond it are retracted with it); and if the new action is in conflict with some action in the tentative context, the conflicting action (and all contributions beyond it) are retracted.

A tentative action stored by one agent A is said to be *in conflict* with the set of actions communicated to it by another agent B if that action is incompatible with those described for the same character in the set of actions already communicated by B, or if it gives rise to tensions or emotional links incompatible with those arising from the same set of actions already communicated by B.

In this model, both agents prepare material in parallel, all the time monitoring the speaker's contribution, and accepting the need to revise their prepared material in the face of conflicts. This constitues a richer model of the process of improvisation than the original basic process.

A further option for enriching the model might be to ensure that each agent operate on different resources or with a different configuration of the engagement-reflection cycle.

Each agent might have different content in their knowledge structures, i.e. the previous stories for each agent can be (either slightly or radically) different. In the same way, the pre and post conditions of story actions may have some differences between agents. This will produce unique contexts, i.e. contexts that do not exist in the agents' knowledge-base and that might lead to interesting plots.

In normal conditions MEXICA evaluates the material generated during engagement each time it switches to reflection. Although MEXICA generates plots through engagement-reflection cycles, the systems is also capable of producing material employing only the engagement routines or only the reflection routines. Thus, each agent can perform in different configurations of the cycle. For example, one agent can perform under engagement only and the other under reflection only, or one under engagement only and the other under engagement & reflection, etc. That is, agents might evaluate coherence, interestingness and novelty at different times during the whole improvisation process. By default MEXICA generates three actions during engagement and then switches to reflection to evaluate the material generated. The system evaluates coherence by checking that all actions' preconditions are fulfilled. If necessary, the system inserts actions to satisfy preconditions. If the system is not able to produce an action during engagement it switches to reflection and inserts one action to continue the story. In this way, one engagement-reflection cycle is completed. The outcome of these processes might range from one to several actions (depending on how many actions are inserted during reflection). Thus, during improvisation both agents run in parallel one engagement-reflection cycle. In this way, one agent is contributing with the next action in the story and the other one is trying to prepare material in advance, i.e. to anticipate possible directions that the improvisation might take in order to avoid lags. This process changes slightly if one of the agents is running only during the engagement mode or only during the reflection mode.

In this way, we represent the fact that real actors have different knowledge, experiences, perceptions of the world, theatrical resources, etc., and nevertheless they are able to produce an improvisation.

4.2 Interconnecting the Storytellers

In order for our simulation to constitute a plausible model of colaborative storytelling as carried out by humans, it is important that the information shared between the story tellers be restricted to communication acts equivalent to saying out aloud a sentence - or a group of sentences - intended as a contribution to the story so far. For the sake of simplicity of the model, these communication acts may take the form of valid utterances in some formal or semiformal language rather than natural language sentences. This avoids to a certain degree the need to address the problem of natural language understanding, which is known to be complex. We operate under the assumption that the only requisite for our model to be plausible is that whatever form is being used to communicate be easily convertible into the internal representation that the storytellers are using. The OAA's Interagent Communication Language (ICL) constitutes a good vehicle for the type of semiformal communication that is envisaged.

With respect to implementation details, this particular task of the model has not been addressed yet, since most work has focused so far on getting the colaborative storytellers operative. It is our intention to model the actual process of conversation by launching each storyteller as an individual agent within an Open Agent Architecture setting. In this way, the OAA Facilitator would act as mediator between the agents, and the communication protocols provided would guarantee the required level of abstraction from the low level detail of communication between each storytelling process. As an additional advantage, we contemplate the possibility of taking advantage of OAA's functionality to run each story teller on a different machine, and connect them into a distributed network of storytellers.

5 Discusion

An important issue to consider is whether the proposed solution differs in any significant way from the simpler turn-taking version where each participant only starts preparing his contribution once other players have finished theirs.

This needs to be addressed at three different levels. On one hand, it is expected that the proposed solution would show an improvement in efficiency, reducing possible delays between the contributions of different speakers. However, the response times of the story generators for preparing tasks involving the type of short contributions envisaged here may be too short for any significant difference to become apparent.

On the other hand, for similar response times the quality of the contributions must be considered. If a player has been lucky and no conflicts have arisen, he can start straight away presenting a more complex contribution than he might have put together if he started preparing from scratch. If severe conflicts have appeared, severe enough to force complete rejection of everything prepared so far, the speaker would not be worse off than if he had not prepared at all. There is an intermediate possibility, where only part of the plan needs to be rejected. This situation still leaves the speaker slightly ahead of the game, since he already has some material to kick-start his contribution. Additionally, for automatic storytellers this option has the advantage of introducing a factor of variation: the contributions that may result from building upon part of a previous plan that has had to be pruned may be different from what would have been planned from scratch.

Regarding the improvisation issues that we identified earlier as key elements for this research, we would like to mention a few relevant questions. An improvisation might be considered "good" when: it is interesting, coherent and novel; and it is the result of the interaction of at least two independent agents with different content in their knowledge-bases and/or different operation modes. MEXICA provides the mechanisms to satisfy this requirement. On one hand, we are employing MEXICA's methods to evaluate the interestingness, coherence, and novelty of a story. In MEXICA a story is interesting when it includes increments and decrements of the tension; a story is considered as novel when it is not similar to any of the previous stories in its knowledge-base; and it is considered coherent when all actions' preconditions are fulfilled within the story. On the other hand, each MEXICA agent can have different knowledge since their knowledge structures are created from the files of previous stories provided by the user. So, if they are different, the atoms for each agent are different. MEXICA can work in four different operations modes; furthermore, the system includes more than 20 parameters that control different functions within system. Thus, the behavior of each MEXICA agent can be controlled by the user. If we build agents with different knowledge and behavior we avoid developing a simple turn-taking computer program. Coherence in improvisation is a very complex problem. The current version of MEXICA handles the coherence issue by modifying the material previously generated, which is not an option for improvisation. So, we require to develop new routines that help us to deal with this situation. But at least MEXICA is able to point out problems of coherence. Regarding the representation of different knowledge and experiences, in MEXICA each character has its own representation of the story-world context. So, employing the same structures each actor during improvisation can have its own representation of the word. Experiments will tell us if that is enough to generate good improvisations. Concerning emergent contingencies, when tensions like love competition or clashing emotions arise in a story, there is a good opportunity to create interesting plots. MEXICA already is capable of detecting and exploiting these situations. With respect to shared representations of the world, MEXICA employs clusters of emotional links and tensions between characters, referred to as contexts, to represent the state of affairs of the story-world. As mentioned earlier, contexts are very dynamic structures that can be easily built and modified during improvisation. So, we believe they can nicely support the representation of the new story-world created during improvisation.

The architecture described in this paper constitutes a very interesting platform from which to address some of the more mistifying issues of improvisional creativity. It may seem that setting two programs against one another may reduce the interest of the experiment to whatever restricted capabilities the automatic storytellers can model, which need not be as many or as good as a human storyteller might have shown. The option of connecting them via a multiagent architecture leaves open the possibility of developing an interface module for humans to participate in the task. In a kind of Wizard of Oz experiment, the human would initially attempt to reproduce the behaviour of an automatic storyteller, as specified in their description. However, if the interaction is recorded, all departures by the human from the specified behaviour would be duly noted, and this record could then be carefully studied to identify functionalities that might improve the performance of the automated storytellers.

An important issue is how the system would scale if more agents are used. It seems aparent that the introduction of more than one agent may result in an improvement of the variety of the resulting stories, as a result of a greater range of possible contexts being taken into account when building the story. However, each increase in the number of agents also increases the risk of fruitless computations (those that give rise to tentative actions that are later rejected either due to conflict or redundancy). A balance must be sought between the added variety and the loss of efficiency introduced by collaboration.

6 Conclusions and Further Work

An interesting issue that may need to be considered in further work is whether the actors may also use preparation or planning at a more abstract level, to introduce in the story material that they hope to be able to use at later stages. This would correspond to extending the idea of a tentative context so that agents may maintain actions as tentative even while they are operating in speaker mode, in the hope of contributing them during a later turn. This opens interesting possibilities, both in terms of how that preparation might take place and how it interacts with short-term preparation. Additionally, there is a revision problem equivalent to the one occurring for preparation, in as much as the material introduced by one player with a particular aim in mind may be exploited by the other in a different, possibly conflicting way. This would force the original player to forsake his long term plan, or at least to modify it.

Existing academic work on oral literature may provide keys to techniques and resources that human storytellers have used in the past to solve problems of lack of inspiration. These include the use of formulaic constructions to resolve descriptions of characters, locations or events, the insertion of brief messages addressed to the listeners - possibly intended to build suspense or to draw attention to particular ingredients of the story -, or the introduction of parallel stories as subtexts. Such resources may be considered as possible expansions of the architecture presented here if it is considered that their addition may improve the quality of the resulting story. They may provide a good way of covering up any noticeable gaps in the sequence when radical revision of prior preparation forced by conflict leaves a speaker with no material to start contributing immediately.

Acknowledgements

This work is partially supported by the Spanish Ministry of Education and Science project TIN2006-14433-C02-01, and research group grant UCM-CAM-910494, jointly funded by Universidad Complutense de Madrid and the Comunidad Autónoma de Madrid (Dirección General de Universidades e Investigación).

References

Agre, P. (1997). *Computation and Human Experience*. Cambridge University Press, Cambridge.

Agre, P. and Chapman, D. (1987). Pengi, an implementation of a theory of activity. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, Seattle.

Bailey, P. (1999). Searching for storiness: Story-generation from a reader's perspective. In *Working Notes of the Narrative Intelligence Symposium, AAAI Fall Symposium Series.*, Menlo Park, CA. AAAI Press.

Bellinger, M. F. (1927). *A Short History of the Drama*. Henry Holt and Company, New York.

Cheyer, A. and Martin, D. (2001). The Open Agent Architecture. *Journal of Autonomous Agents and Multi-Agent Systems*, 4(1):143–148.

Meehan, J. R. (1977). Tale-spin, an interactive program that writes stories. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, Cambridge, Mass. Morgan Kaufmann.

Moraes, M. C. and da Rocha Costa, A. C. (2002). How planning becomes improvisation? - a constraint based approach for director agents in improvisational systems. In *Advances in Artificial Intelligence, 16th Brazilian Symposium on Artificial Intelligence, SBIA 2002*, volume LNCS 2507, pages 97–107.

Pérez y Pérez, R. and Sharples, M. (2001). Mexica: a computer model of a cognitive account of creative writing. *Journal of Experimental and Theoretical Artificial Intelligence*, 13(2).

Rumelhart, D. (1975). Notes on a schema for stories. In Bobrow, D. G., editor, *Representation and Understanding. Studies in Cognitive Science*, New York. Academic Press.

Sharples, M. (1999). *How We Write: An Account of Writing as Creative Design*. Routledge.

Trastoy, B. (2005). La improvisación: estrategias productivas de una práctica escénica alternativa. hablan los teatristas. *Telondefondo*, (1).

Turner, S. R. (1994). *The Creative Process: A Computer Model of Storytelling*. Lawrence Erlbaum, Hillsdale, NJ.

Narrative Inspiration: Using Case Based Problem Solving to Support Emergent Story Generation

Ivo Swartjes, Joost Vromen and Niels Bloom
Human Media Interaction
University of Twente
PO Box 217, 7500 AE Enschede, The Netherlands
{swartjes,vromen,bloom}@cs.utwente.nl

Abstract

We consider a system that can generate stories as a creative system. One approach to building such a system is to simulate and narrate the behaviour of believable characters in a virtual story world. A risk of this approach is that no interesting story emerges. In order to make the behaviour of the characters more interesting from a story perspective, we propose a system that can use example story pieces, written from a plot perspective by a human author, to inspire decisions for characters in an emerging story.

On a more philosophical note, we discuss the story generation process in the light of a characterization of creative systems and show that considering an automated story generator as a creative system can help to reveal implicit design choices.

Keywords: Story Generation, Computational Creativity, Case Based Reasoning, Fabula.

1 Introduction

The Virtual Storyteller is an experiment in the creation of stories through simulation of a dramatic story world inhabited by virtual characters (Theune et al., 2004). We separate the content of a story from its presentation (e.g., in the form of natural language or animation). The focus of this paper will be on the generation of story *content*, meaning we focus on a way to generate an interesting sequence of events.

The characters in the Virtual Storyteller are modelled to be believable, which means amongst other things that they are pursuing their own goals, have an emotional model and a simple form of personality[1]. Believable behaviour does not necessarily lead to a coherent and interesting plot, though. This was the big lesson learned from one of the first systems to use character models to generate stories, namely TALE-SPIN (Meehan, 1981). TALE-SPIN was able to tell simple Aesop-like stories about animal characters that were trying to fulfil their basic needs. The stories generated by TALE-SPIN were certainly believable but often uninteresting. There is no guarantee that the coincidental interaction between the characters results in a dramatically interesting and coherent whole. Subsequent approaches to story generation have therefore tried to focus more on plot development. In these approaches, the succession of a sequence of dramatic events is modelled, and characters are placed in function of these events (*right...we need a robbery...which characters can we use for that?*) which in turn makes it quite difficult to have the characters appear believable.

A contemporary example of character-driven narrative is the FearNot! system, which simulates affectively driven characters that exhibit and respond to bullying behaviour (Aylett et al., 2005). The goal of the system is to educate children – who play the role of invisible friend of the main character that is being bullied – how to deal with such behaviour. A typical plot-driven approach is the Fabulist system (Riedl and Young, 2005), which is able to plan a plot that fulfils certain dramatic goals, whilst trying to relate the plan steps to character intentions and personality.

We are investigating character-driven (emergent) story generation as in the FearNot! project. The main difference is that in the approach we pursue, characters have a double role: they are believable inhabitants of a virtual story world as well as improvisational actors that help to create an entertaining experience. With the latter, we hope to make up for the potential lack of story development when using only believable characters. We also intend to use a plot agent that influences the course of the story to increase the chance of interesting plot developments.

In the course of an emergent narrative, there are many decisions to make for both the character agents (in terms of which goals to pursue, how to respond emotionally to new information) and the plot agent (in terms of how to affect the emerging story). Character agents are implemented using an affective architecture with the aim of making believable decisions. We hope to augment the decision space with options that are also interesting from a story development perspective. We want to allow human

[1]An extensive discussion of what it means for virtual characters to be believable can be found in (Loyall, 1997)

authors to be able to write example story pieces and allow a Case Based Reasoning (CBR) system to use these pieces as example solutions that inspire some of these decisions.

After discussing some work related to the use of CBR in story generation in section 2, we will describe a creative problem solver based on CBR and explain how it can inspire decision making for the character and plot agents in section 3. Sections 4 and 5 will provide a bit more in-depth description of the proposed system in terms of cases and creativity heuristics, respectively. The implementation of the creative problem solver is discussed in section 6. Section 7 will position the design of a story generator within the context of the formal framework for categorizing creative systems proposed by Wiggins (2001). We argue that such a positioning is useful to make more informed choices in the design of creative story generation systems.

2 CBR in the Context of Story Generation

Case Based Reasoning (CBR) is a reasoning process that finds its origins in the psychological model of episodic memory. In CBR, knowledge is captured in the form of a set of cases that describe a specific problem, and a specific solution to that problem. To reason about solutions for problem situations, the collection of cases is explored in order to find similar situations. The solutions for those similar problem situations can be adapted to form a solution to the problem that needs to be solved.

CBR has been applied in the context of storytelling or storytelling-like systems with satisfying results (Mueller, 1987; Turner, 1994; Fairclough, 2004; Gervás et al., 2004). With the exception of Mueller (1987), who uses CBR in a system that generates daydreams, the use of CBR in these systems has been inspired by or directly based on the work of Vladimir Propp (Propp, 1968). Propp analysed Russian folk tales, and discovered a big structural similarity between them. He identified a set of character roles (e.g., the hero, the villain) and a set of character functions (e.g., departure, interdiction and marriage). Each of the folk tales he investigated contains a subset of the character functions, and always in a certain order. This makes Propp's analysis pleasant to formalize and use in story generators, but its very order constraints make it difficult to use in the emergent story generation process we are exploring, where – theoretically – anything can happen that does not conform with this order.

We therefore use CBR to generate high-level character behaviour based on story-specific input rather than Proppian plot variations. We want to provide a human author with the possibility to write story content that has the flexibility to be recombined and reformed to expand the space of situations in which it can be used. Our problem solver, discussed in section 3, has been influenced most strongly by the MINSTREL system (Turner, 1994). Turner considers creativity to be an extension of problem solving, the result of cognitive processes that bring together pieces of old knowledge in new ways. MINSTREL has demonstrated the possibilities of case based problem solving to model the creative process of generating simple stories in the King Arthur domain. MINSTREL takes storytelling goals as problems to solve, and retrieves cases that form a solution to these problems. Usually, CBR retrieves solutions by comparing the problem to solve with similar problems in the case base. MINSTREL adds creativity to this problem solving by actively transforming the problem descriptions into similar descriptions, and subsequently adapting the found cases to fit the original problem. MINSTREL uses a collection of Transform-Recall-Adapt methods (TRAMs) for this process.

For example, if MINSTREL attempts to create a story in which a knight commits suicide, a problem description could express the following problem: *A knight does something that results in the knight's death.* A TRAM can transform this problem description into: *A knight does something that results in someone's death.* Based on this problem description, the following case can be retrieved:

> *A knight fights a troll with his sword, killing the troll and being injured in the process.*

This retrieved case can then be adapted to form a solution to the original problem description:

> *A knight fights and kills himself with his sword.*

A combination of such problem solving steps, guided by author-level goals and themes, leads to the structural composition of simple stories.

MINSTREL was implemented using a custom frame-based language called Rhapsody, implemented in Lisp. Currently, attempts are being undertaken to re-implement MINSTREL using the contemporary knowledge formalism OWL-DL, a dialect of the well known ontology language OWL resembling Description Logic (Peinado and Gervás, 2006). Similarly, our problem solver uses knowledge from a story world ontology specified in OWL-DL for its transformations. The problem solving cycle has big similarities to MINSTREL's model of creativity but is only loosely based on its specific details. As we will see in section 3, we intend to use case based problem solving for a different purpose than the structural composition of a story from cases, as done in MINSTREL.

3 Using Case Based Problem Solving for Character and Plot Decisions

Our problem solver uses CBR for two reasons. First, because the knowledge needed for a certain problem domain is covered by a set of example cases instead of an extensive set of 'first principles'[2], CBR can decrease the amount of knowledge needed and reduce or eliminate the need to model the causal interactions of this knowledge in the form of a reasoning system (Cunningham, 1998).

Second, we believe that cases form an intuitive way for a human author to write story content. As we will discuss in section 7, the designer of a story generator is also responsible for the quality of its generated stories. It therefore makes sense to make the input knowledge as

[2]In the case of our decision making, these are principles like 'a person can fight a dragon when he is near one, has a weapon and is not too afraid' and 'if you are hungry, and you know where food is, then you should eat the food'.

accessible to authors as possible and we believe that using examples (rather than worrying about the 'first principles' rules that underly them) satisfies that concern.

The cases in our problem solver therefore take on the form of example pieces of story. We believe that such example story pieces can form a good knowledge source in the decision making of character agents in a storytelling domain, most importantly because they describe character behaviour in a narrative context, transcending individual decision making. Take for example a case expressing the following example story piece: *Frustrated by her singing, John insults his little sister, making her cry*. Not only does this case offer a character a believable coping behaviour for being frustrated at one's sister, it is also an *interesting* decision because it affords an interesting story situation (in this case, a decision that affects important other characters). When the cases have been constructed to express believable character behaviour as well as interesting narrative situations, using these knowledge sources results in behaviour that is in theory both believable and interesting.

The character agents of the Virtual Storyteller make decisions based on appraisal and deliberation processes (which goals to take on, which actions to pursue, how to interpret perceptions and how to respond to them emotionally). As an alternative to using these processes, such decisions can be contracted out to the creative problem solver by translating them into problem descriptions and asking the problem solver to find solutions for them. We aim for an integration of this decision making within the processes that our character agents already run, similar to the work of Moraes and Costa (2004) which shows how to make such an integration of 'improvised' decision making within a BDI architecture.

The cases can also be used by the plot agent to make decisions about influencing the emerging story. Such decisions involve introducing dramatic events, adding new knowledge about the story world, or suggesting actions and goals to the characters. Because the plot agent uses much of the same episodic knowledge as the characters, it can make reasonable assumptions about the character's reaction to these events. In-depth discussion of the integration of the decision making process with the Virtual Storyteller architecture falls outside the scope of this paper, which focuses rather on the decision making process itself.

Figure 1 shows the decision making process we propose, including the CBR problem solving cycle we intend to incorporate in the Virtual Storyteller. A user – be it one of the character agents or the plot agent – needs to make a decision which it translates into a problem for the creative problem solver. The problem solving cycle starts, in which copies of the problem are transformed into similar problems. This is a recursive process; transformed problems can again be fed into the problem solving cycle. Cases that match the transformed problem specification are retrieved, and adapted so that they form a solution to the original problem. This will be explained in more detail in sections 4, 5 and 6.

The choice which of the creative solutions to use as decisions for the users of the creative problem solver (i.e., the character and plot agents) should be informed by con-

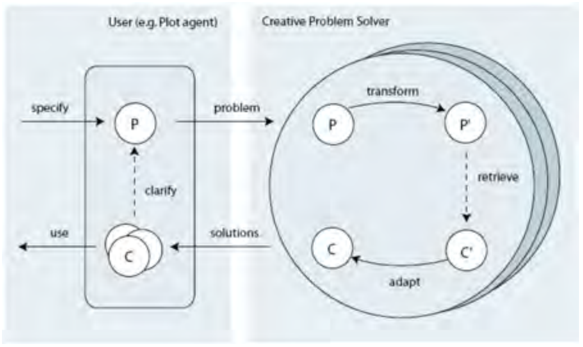


Figure 1: The process of decision making using the creative problem solver

straints given by the users. Inherent to unconstrained creativity is that creativity errors can occur. A child having a limited concept of objects and holes could have a creative idea to fit a square peg through a round hole. The idea is understandable, but reality proves that the solution does not work. By constraining the valid solutions, we can decrease the number of creativity errors. These constraints could be given by a domain-specific model of the 'impossibilities' of the domain or by the state of the story world at a particular moment in time. If a solution meets the constraints, it can be used as a decision. If not, the problem can be *clarified* by extending it with extra information in the form of new constraints. For instance, it could be that a solution for the knight to use a sword to kill himself does not work, because the story world contains no sword. The problem description should at that point be clarified: 'find solutions that do not contain a sword'.

4 Knowledge Representation

A case in the context of our system expresses an example story piece in a formalized language. As discussed before, this piece should express both believable behaviour of the character(s) partaking in it, and the narrative context of the event sequence itself. We will first discuss a knowledge representation used to express such story pieces, and then give formal definitions of case and problem representations.

4.1 Fabula Representation

In the ability to interpret a sequence of events as a story, causality between events plays a major role (Trabasso and Nickels, 1992). Indeed, the story generation system MAKEBELIEVE (Liu and Singh, 2002) is able to create simple stories using only facts about causality between situations as represented in a large common sense knowledge base. From the viewpoint of narratology, a distinction is often made between the *fabula* of the story, a series of causally and chronologically related events that are caused or experienced by characters in a story world, and the *sjuzet*, a dramatic and subjectified abstraction of the fabula.

The Virtual Storyteller produces fabula which is subsequently fed to processes that select and narrate a sjuzet.

We capture the fabula of our simulated story world in the form of a knowledge representation that is discussed in Swartjes and Theune (2006). This representation captures the temporal-causal course of events in the story world. The representation is given by a quadruple $< E, T, C, D >$ where E is a set of fabula elements, T is a set of temporal annotations to these fabula elements, C is a set of causal relationships between fabula elements, and D is a set of descriptive contexts that are linked to the fabula elements and describe their contents. E is divided into six categories: Event, Perception, Internal Element, Goal, Action and Outcome. C is divided into four categories: physical causality, psychological causality, motivation and enablement. Elements of D are subgraphs that can contain fabula as well; this allows for embedded expressions like:

> "The bank owner believes *that it is the bank robber's goal to be rich, and that that goal motivates an action for the robber to rob the bank*; this belief psychologically causes the bank owner to be scared."

The fabula representation is based on a cognitive model for the comprehension of simple stories (Trabasso and Nickels, 1992). This model describes the causal connections that children ascribe to a series of events in a picture story in their attempt to understand the story that the picture sequence conveys. The Virtual Storyteller generates similar causal connections; a fabula is the result of logging the causality between dramatic events that happen in the story world, the resulting beliefs and emotions of the character agents, their goals, attempts to reach these goals in the form of planned actions, and the outcomes of these goals once a goal is achieved or abandoned.

4.2 Problem and Case Representation

For a smooth integration of the problem solver with the Virtual Storyteller, the fabula representation is used to express both the cases in the case base of the creative problem solver, and the problems that should be solved. The problems can be constructed by the agents that use the problem solver; the cases are in principle constructed by a human author.

A problem P is a tuple $< Pat, Con >$ where Pat is the pattern space defining knowledge that must occur in the solution, and Con defines the constraint space in the form of a pattern that should *not* occur in the solution. Both Pat and Con are expressed in terms of fabula but may contain uninstantiated elements.

A case C, also expressed in terms of fabula, has the following requirements:

- C demonstrates a *narrative concept*. A narrative concept could for instance be 'hiding from a threat' or 'flying over an area to search for something'. The expression of a narrative concept is an implicit description of an example problem and a solution to it. The problem is for instance the threat, and the solution is to hide. Of course, there can be many cases demonstrating the same narrative concept.
- C is *context complete* with regard to its narrative concept. Cunningham (1998) states that the case representation must capture the predictive features of a problem. Applied to storytelling problems, we define a context complete case as a case that contains all the elements that are necessary for the case to be viewed as 'believable' by the author of the case, regarding the narrative concept it is supposed to express, and contains nothing more than that.

5 Creativity Heuristics

Unlike in standard CBR, MINSTREL uses creativity heuristics that actively transform problems to find cases instead of finding and adapting cases based on similarity metrics. The reasons for this given by Turner are that truly creative solutions are not found if the problem stays intact, and that adaptation of retrieved cases to fit the problem specification is very difficult. By using transformations and a simpler retrieval mechanism, the adaptation of retrieved cases can be done by reverse application of the transformations.

To guide the transformation of the problem space, we use creativity heuristics similar to Turner's TRAMs. The heuristics are domain-specific and provide a way to transform a problem P into a similar problem P'. When P' enables the system to retrieve a case C, the used heuristic defines a way to apply a reverse transformation to C to create C' which forms a solution to P. When searching for a solution, the creative problem solver can use any applicable creativity heuristic to transform the problem. Problems can undergo a series of these transformations in succession, although too many transformations will make the problem too dissimilar from the original problem and cause found solutions to be unsuitable to solve the original problem. Therefore, the number of successive transformations are limited.

Our creativity heuristics implement the following steps:

Match determines if the heuristic is applicable to the current problem;

Transform defines how the problem space is transformed;

Retrieve finds the cases that match the transformed problem;

Adapt defines how a retrieved case is adapted to the original problem by applying the transformation in reverse.

Turner (1994) has implemented and evaluated a number of creativity heuristics in MINSTREL. Most of them can be divided into a number of functional categories: relaxation, generalization, substitution of a similar subpart and planning knowledge. We will show how we have adapted two of MINSTREL's heuristics to the problem solving domain of the Virtual Storyteller: Generalize Actor and Switch Intention. We will discuss these heuristics below.

5.1 Generalize Actor

Some of MINSTREL's most useful heuristics involve generalization: moving from a specific problem to a more general problem. Elements that can be generalized in our fabula representation are for instance objects in the story world, character roles, actions and their actors, represented in terms of domain specific OWL ontologies. Generalization heuristics assume that a problem definition will remain valid when one such element is replaced by a generalization of that element. An example is shown for generalization of the actor of an action (generalizing other elements proceeds in a similar fashion):

Match: The problem should contain at least one action individual with an `agens` relation to an actor individual. The `agens` property of an action refers to the character performing the action.

Transform: Select one such actor individual at random, and replace its type by a generalized type based on its ontology hierarchy. Only generalize a type one step up in the hierarchy.

Retrieve: For each of the cases in the case base, check if part of the case description unifies with the pattern space Pat of the transformed problem description, and the case description does not contain knowledge specified in the constraint space. Select the cases for which this holds.

Adapt: Identify the actor individuals from the retrieved case whose type matches the generalized type created by the Transform step. Replace all such actor individuals and their type with the actor individual and type from the original problem.

Consider a fragment of a problem description expressing "A princess runs away from a dragon." Examples of actor and action generalizations of this fragment (generalized element in italics):

1. "A princess runs away from *a monster*," leading to the retrieval of cases about princesses running away from orcs and trolls;

2. "*A woman* runs away from a dragon," leading to the retrieval of cases about queens, shepherdesses or little girls running away from dragons.

3. "A princess *moves* away from a dragon," leading to the retrieval of cases about princesses walking, sailing or swimming away from dragons;

5.2 Switch Intention

Switch Intention is an example of a heuristic that substitutes a subpart of a problem for a subpart that is similar in meaning. Switch Intention is based on the idea that if something happens unintentionally, one can try to make this happen intentionally. Actions can cause events that the agent did not intend or expect, or failed to take into account. Cases describing this can be transformed into cases where these events *are* intended:

Match: The problem should contain at least one goal - uninstantiated action - positive outcome combination. In other words, the problem is about "what action brings the goal to a successful outcome?"

Transform: Using the goal found in the match step, find an event that achieves that goal[3]. Construct a new problem containing an uninstantiated action that unintentionally causes the found event. The new problem is about "what action can cause an (accidental) event that brings the goal to a successful outcome?"

Retrieve: Similar to the Retrieve step of the Generalize Actor heuristic.

Adapt: Replace the original uninstantiated action with the action found in the retrieved case, and add the fact that the original goal (from the match step) motivates this action.

This heuristic could be used in a context like the following. A little princess wants to go play outside but needs to ask the king for permission. When she finds him, he is sound asleep in his chair. What can the princess do to wake up the king? The Switch Intention heuristic is based on the premise that instead of executing an action to wake up the king in some way, maybe she can cause an event that wakes him up. If the case base contains a case where a burglar wants to close the door and therefore slams it shut, accidentally waking up the house owner, this case can be transformed using the Switch Intention heuristic so that the princess will slam the door shut in order to wake up the king.

6 Implementation

A prototype implementation has been developed in Java. The ontolologies of the fabula and the objects in the story world are expressed in OWL-DL. At the moment, these are simple ontologies that do not use the full extent of OWL-DL's expressive power. The fabula representation used to describe the cases and problems are expressed as named RDF graphs to express modality (Carroll et al., 2005).

An RDF graph is a set of triples $< S, P, O >$ expressing subject, predicate and object of a fact, respectively. A named RDF graph is an RDF graph with an identifier that can be referred to. Each triple that is part of a named RDF graph can be described as a quadruple (called *quad*) $< G, S, P, O >$, with G being the graph identifier. This contextualizes statements and allows for expressing information *about* triples by referring to their graph identifier. So not only can we express that a certain goal to attain some state motivates an action to walk somewhere:

```
(maingraph
     goal.2 motivates action.9)
(maingraph
     goal.2 rdf:type AttainGoal)
(maingraph
     action.9 rdf:type Walk)
```

[3]This follows from the effects of the events if the events are represented as planning operators.

```
...
```

but we can also express what the goal *means* (for instance, the goal is that the princess has a friend):

```
(maingraph
     goal.2 hasContents graph.5)
(graph.5
     princess.1 hasFriend friend.5)
...
```

The prototype implementation is based on Jena[4] and uses a process of querying (using the SPARQL query language[5]) and transformation (using low-level quad replacement) of these sets of quads. The match step queries the pattern space of the problem description, and one result of this query is chosen at random. This result is a set of variable bindings; the quads that were used to bind these variables are mapped to their transformed counterparts (e.g., by looking up the superclass of an individual's type). The mapping is remembered for the adaptation step. In this step, the original quads are used to replace quads from the retrieved case by quads with their original values filled in.

We have implemented the problem solving cycle using standard breadth-first search, assuming that solutions that are not found with a limited number of successive transformations, will be too far off to form a good solution to the problem being solved. Preliminary results indicate that even with two generalization heuristics and a limit on the number of transformations, the search space quickly becomes quite big, due to the fact that there are many ways in which one heuristic may match the problem description. Even a simple problem description may contain quite a number of objects and actions that can be generalized. For application in the Virtual Storyteller in reasonable processing time, we might have to make concessions in the number of calls to the creative problem solver and the creative possibilities of it. Another option to limit the processing time is to pre-process transformations of some expected problem descriptions offline.

7 Story Generators as Creative Systems

Wiggins (2001) discusses a formal framework to define and categorize creative systems. Based on the work of Boden (1990), Wiggins discerns the ingredients of an exploratory creative system (i.e., a system that selects and values partial or complete concepts that are found by traversing a conceptual space) and considers transformational creativity (i.e., creativity that changes the rules which define this conceptual space) as exploratory creativity at the meta-level. Placing systems that can automatically generate stories within this framework explicates design choices that are sometimes made implicitly. In the discussion of automated story generation as a creative process, the following terms are relevant:

- $\mathcal{C}$: the conceptual space, which in the case of a story generator can be interpreted as the set of "well-formed stories" for a given domain.
- $\mathcal{R}$: the constraints that define $\mathcal{C}$, which can be interpreted as the rules that determine whether a potential story is well-formed.
- $\mathcal{T}$: the rules that specify how to traverse the conceptual space, which can be seen as the story generation algorithm.
- $\mathcal{E}$: the constraints that evaluate $\mathcal{C}$, which can be seen as the rules that determine the quality of the story.

Riedl and Young (2005) discuss story planning in the context of exploratory creativity. They claim that the evaluation criteria $\mathcal{E}$ are not generally known or knowable in the domain of storytelling. The rules that constitute $\mathcal{E}$ (i.e., define the quality of a story) seem indeed difficult to formalize. This is why a creative story generation system must somehow be set up in such a way that it does not rely on evaluation by $\mathcal{E}$. The implication is that the system must deliver good stories without having knowledge about *why* they are good. The input of the system should already be fertile with narrative potential and the developer of the system therefore also becomes, in a sense, responsible for the quality of the produced stories. Taking this position emphasizes the role of authoring: an effective story generation process should be transparent enough for the developer to have an understanding of the relationship between certain input and their effect on the generated stories.

However, what *is* possible to some extent is to determine $\mathcal{R}$, i.e., the "well-formedness" of the generated stories. This at least allows a story generator to explore a space of possible stories. Story generation systems often use formalizations of $\mathcal{R}$ based on findings in narratology (e.g., Propp) or story understanding (e.g., story grammars as used by Lang (1999)) . The Fabulist system (Riedl and Young, 2005) is based on two criteria: character believability and plot coherence. The first criterium (say, $\mathcal{R}_1$) is formalized by requiring that every action in the story is intended by a character; the second criterium (say, $\mathcal{R}_2$) is formalized by requiring that every action has a direct or indirect causal relation to the outcome of the story. Such a formalized $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$ enables the traversal of the conceptual space by means of a story planner $\mathcal{T}$. The fulfilment of $\mathcal{R}_2$ is a direct result of using a Partial Order, Causal Link planner (POCL) which starts its planning process from the outcome of the story and plans its way back satisfying causal requirements. Such a planner would never incorporate plan steps that have no causal relationship with the outcome.

Our approach to story generation can be viewed as a process of *exploratory* creativity. We adopt criteria similar to Riedl and Young (2005) for the well-formedness of a story, and traverse the conceptual space by using autonomous characters that are designed to meet $\mathcal{R}_1$, i.e., to be believable. We hope to meet $\mathcal{R}_2$ by equating the outcome of the story with the outcome of a particular important goal of one of the characters. Our approach does not ensure that $\mathcal{R}_2$ is met, since it is easy to imagine other characters doing things that have no relevance whatsoever to the outcome of the chosen goal. But if a certain story does not meet $\mathcal{R}_2$ (i.e., the story contains parts that have no causal relation to the outcome), certainly we can

[4]http://jena.sourceforge.net

[5]See http://www.w3.org/TR/rdf-sparql-query/

find a subset within the event sequence that *does* meet $\mathcal{R}_2$ (namely, only those parts that *are* causally related to the outcome of a certain goal), as long as we make this causality explicit. For a more detailed explanation of these considerations, see (Swartjes and Theune, 2006).

7.1 The Creative Problem Solver as Creative System

Our creative problem solver is a creative system in its own right, performing a process of *transformational* creativity. In this case, we have different interpretations of Wiggins' terms:

- $\mathcal{C}$: the conceptual space, being the set of solutions in terms of narrative cases.
- $\mathcal{R}$: the constraints that define which solutions are in $\mathcal{C}$, which can be interpreted as the problem description.
- $\mathcal{T}$: the rules that specify how to traverse the conceptual space, which in this case is simply the retrieval query.
- $\mathcal{E}$: the constraints that evaluate $\mathcal{C}$, which can be seen as the rules that determine the quality of the found solutions.

A problem description $\mathcal{R}$ defines what a concept case c should look like (thus being a constraint on the conceptual space $\mathcal{C}$). We then transform this problem description into a different one $\mathcal{R}'$ on the assumption that it is a similar problem description. Effectively we have transformed the rule that determines what concept cases are in $\mathcal{C}$. This makes the process one of transformational creativity. And this process of transformational creativity is indeed an exploratory creativity process at the meta-level if we consider the problem space $\mathcal{R}$ to be the conceptual space of this meta-creative system: we use creativity heuristics – the rules of the transformational creativity – to explore the set of possible rule sets. We enter difficult terrain here as neither the rules that select appropriate problem descriptions nor the evaluation criteria for these problem descriptions being valued are easy to formalize. We make a (possibly invalid) assumption that if a problem description $\mathcal{R}$ is appropriate, then the transformed problem description $\mathcal{R}'$ is also appropriate, by carefully defining the creativity heuristics. This explicit design effort is in line with Wiggins' view on transformational creativity, where the designer needs to be aware of the rules being applied rather than rely on serendipity (Wiggins, 2001).

8 Conclusion

The Virtual Storyteller generates stories based on two criteria for their well-formedness: they must portray believable character behaviour and contain coherent plots. Believable character behaviour is achieved by simulating a story world in which autonomous character agents try to achieve their goals and respond emotionally to their environment. A coherent plot is formed by a subset of the fabula generated by the simulation that contains a coherent causal chain (e.g., a chain of events that have contributed to the outcome of an important goal of one of the characters).

We extend the design of the Virtual Storyteller with a creative problem solver that allows for decision making that falls outside the range of decisions that would have been made using only autonomous character agents. The problem solver uses example story pieces that can provide solutions to decision problems that occur in the course of the simulation.

Adding the creative problem solver to the Virtual Storyteller addresses two issues. First of all, a known risk of using autonomous character behaviour to generate stories is that no interesting story emerges. We think that example story pieces, written from a story perspective, allow for more interesting stories to happen, since they define character behaviour in a story context that transcends the decision space of the characters' individual behaviour. Second, the designer of the story generation system is in a sense responsible for the production of good stories, since it is difficult to formalize rules that assess the quality of a story. This stresses the importance of authoring: the architectures and input knowledge of the story generator should afford the designer-as-author to express his narrative intent. We believe that example story pieces form an intuitive way to write such input knowledge.

The creative problem solver performs a process of transformational creativity to extend the range of use of these story pieces, by transforming input problems into problems that are similar in meaning, and adapting found cases to provide a solution to the original problems. This process is guided by explicitly defined creativity heuristics. However, the use of such explicitly defined heuristics does not guarantee finding correct solutions. For instance, the assumption that limiting the number of transformations keeps the solution applicable for the problem at hand, might fail. If the concept of 'dragon' is generalized to 'organism', the system might come up with a creative solution where a knight eats a princess to satisfy his hunger, because it uses a case where a dragon did the same. Evaluating the solutions and tuning the ontologies, heuristics and search control to decrease errors is therefore important. We have designed and partially implemented the problem solver; future work will involve further implementation and evaluation of the problem solver in the context of the Virtual Storyteller.

Acknowledgements

This work has been funded by the Strategic Research Orientation NICE of CTIT, the Netherlands.

References

Aylett, R., Louchart, S., Dias, J., Paiva, A., and Vala, M. (2005). FearNot! - an experiment in emergent narrative. In *Proceedings of the 5th International Workshop on Intelligent Virtual Agents*, pages 305–316.

Boden, M. (1990). *The Creative Mind: Myths and Mechanisms*. Abacus, London.

Carroll, J. J., Bizer, C., Hayes, P., and Stickler, P. (2005). Named graphs. *Journal of Web Semantics*, 3(4).

Cunningham, P. (1998). CBR: strengths and weaknesses. Technical report, University of Dublin, Computer Science Department.

Fairclough, C.R. Cunningham, P. (2004). AI structuralist storytelling in computer games. Technical report, University of Dublin, Computer Science Department.

Gervás, P., Díaz-Agudo, B., Peinado, F., and Hervás, R. (2004). Story plot generation based on CBR. *Knowledge-Based Systems*, 18(4-5):235–242.

Lang, R. R. (1999). A declarative model for simple narratives. In *Proceedings of the AAAI Fall Symposium on Narrative Intelligence*.

Liu, H. and Singh, P. (2002). MAKEBELIEVE: Using commonsense knowledge to generate stories. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI 2002)*.

Loyall, A. B. (1997). *Believable Agents: Building Interactive Personalities*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA.

Meehan, J. (1981). TALE-SPIN. In Schank, R. and Riesbeck, K., editors, *Inside computer understanding - five programs plus miniatures*, pages 197–226. Lawrence Erlbaum Associates.

Moraes, M. C. and Costa, A. C. d. R. (2004). Imp-BDI: Improvisational BDI architecture. In *Proceedings of the workshop on Architectures and Methodologies for Building Agent-Based Learning Environments*.

Mueller, E. (1987). *Daydreaming and computation: a computer model of everyday creativity, learning, and emotions in the human stream of thought*. PhD thesis, University of California.

Peinado, F. and Gervás, P. (2006). Minstrel reloaded: from the magic of lisp to the formal semantics of OWL. In *Technologies for Interactive Digital Storytelling and Entertainment (TIDSE)*.

Propp, V. (1968). *Morphology of the folktale*. University of Texas Press.

Riedl, M. and Young, M. (2005). Story planning as exploratory creativity: Techniques for expanding the narrative search space. In *Proceedings of the 2005 IJCAI Workshop on Computational Creativity*.

Swartjes, I. and Theune, M. (2006). A Fabula Model for Emergent Narrative. In *Technologies for Interactive Digital Storytelling and Entertainment (TIDSE)*.

Theune, M., Rensen, S., op den Akker, R., Heylen, D., and Nijholt, A. (2004). Emotional characters for automatic plot creation. In *Technologies for Interactive Digital Storytelling and Entertainment (TIDSE)*.

Trabasso, T. and Nickels, M. (1992). The development of goal plans of action in the narration of a picture story. *Discourse Processes*, 15:249–275.

Turner, S. R. (1994). *The creative process: a computer model of storytelling*. Lawrence Erlbaum Associates, Hillsdale, NJ.

Wiggins, G. A. (2001). Towards a more precise characterisation of creativity in AI. In Weber, R. and von Wangenheim, C. G., editors, *Case-Based Reasoning: Papers from the Workshop Programme at ICCBR'01*, pages 113–120, Washington, DC.

Session 2
Analogy & Language

Statistical Evaluation of Process-Centric Computational Creativity

Diarmuid P. O'Donoghue
Department of Computer Science
NUI Maynooth
Co. Kildare
Ireland
`diarmuid.odonoghue@nuim.ie`

Abstract

We adopt a process-centric approach to computational creativity, based on a model of people's innate ability to process analogical comparisons. A three-phase model of analogical reasoning is adapted to function as an analogy generating machine. It is supplied with two distinct knowledge-bases containing many domain descriptions, with the aim of generating novel analogies – potentially even creative ones. However, because our approach to computational creativity does not have the usual "inspiring set", evaluating its output can not be performed by comparison to this inspiring set. Our generic approach to evaluating process-centric computational creativity uses a number of nonparametric statistical techniques. After the creative artefacts are generated, human raters assess these artefacts for the qualities of creativity (*quality, novelty etc*). We describe the results of two experiments that were conducted on these two collections of domains. The analogies generated on the two collections are analysed and difference in the two result sets are assessed. We argue that true creativity can only be assessed after the creative artefacts are generated. Evaluating creativity only by reference to the inspiring set runs the risk of overlooking creative artefacts that differ from the inspiring set - and may under-estimate a model's creativity.

Keywords: Analogical Creativity, Analogy Generation, Evaluation, Nonparametric Statistics

1 Introduction

Computational creativity frequently uses an "inspiring set" of creative artefacts (music, images, poems *etc*) both to drive the model and to act as a basis for its evaluation. This *artefact-centric* approach to computational creativity contrasts with the *process-centric* approach in this paper and elsewhere (O'Donoghue, 1997; Gomes et al., 2003; Veale, 2004; O'Donoghue et al., 2006). This paper describes an approach to computational creativity that is based on people's innate ability to understand analogical comparisons. This paper builds upon computational models of the analogical reasoning process.

Analogical comparisons are often cited as a driving force behind creativity, providing new perspectives on some previously known concepts (Boden, 1992). Creativity using analogies is strongly associated with science and scientific advancement. Pierre Curie and colleagues deliberately used analogies as a technique for generating hypotheses which they later tested (Curie, 1923). Hoffman (1995) and Brown (2003) detail the role that analogies played in many recorded scientific breakthroughs. Dunbar (2001) and Dunbar and Blanchette (2001) note that experts display a great ability to generate and use novel analogies when dealing with situations that arise in their normal work environment (this contrasts with the rare use of analogies by non-experts when presented with tester-determined analogy problems). Koestler (1964) was also among those who account for creativity as the juxtaposition of two very different sets of ideas.

In essence, an analogy is a comparison between two concepts (*source* and *target*) (Gentner, 1983), such that the source highlights particular aspects of the target and suggests new inferences about that target. Every analogical comparison has two effects. Firstly, it highlights an existing non-obvious similarity between two concepts. Secondly, it then extends this similarity by transferring information from one concept to the other, adding new information to the target. In creative analogies (Boden, 1992; Eysenck and Keane, 1995) a strange source domain conjures up a revolutionary new conceptualisation of the target, suggesting inferences that explain some previously unexplained or unnoticed phenomena.

Computational modelling of the analogy process has focused primarily on the central mapping phase (Keane et al., 1994) ; see (French, 2002) for a review). Surprisingly, only a few models have been developed of the previous retrieval phase (Thagard et al., 1990; Forbus et al., 1995; Plate, 1998; O'Donoghue and Crean, 2002; Gomes et al., 2003) or of the subsequent validation phase (Falkenhainer et al., 1989). However, no combined *retrieval–mapping–validation* model has been described and evaluated. In this paper we investigate a three-phase model and evaluate its potential for finding and assessing novel analogies - potentially even identifying creative ones.

We should not be overly proscriptive in how we assess computational creativity, unpredictability being a quality that is often associated with creativity. Any creativity model that is assessed solely by comparison to an inspiring set may inadvertently overlook outputs that are considered creative when assessed independently of that inspiring set. True creativity can only be assessed *post hoc*.

The remainder of this paper is structured as follows. First we review computational creativity, distinguishing between the traditional "inspiring set" approach and our "process-centric" approach. We then describe our computational model for generating and evaluating analogical comparisons. We describe the problem of assessing our model in the absence of an inspiring set. We describe a number of statistical techniques that serve to evaluate process-centric creativity models. We then present and analyse the analogies generated by our model before drawing some final conclusions.

2 Computational Creativity

Ritchie (2001) describes and formalises the typical process by which most computational creativity programs are constructed. The process starts with basic items which are items of the type to be produced by the program (poems, music, images *etc*). A subset of these items is selected, taking into account the ratings and values associate with the basic items - creating the *inspiring set*. Following this, the program is constructed and executed for a range of parameters. We characterise this approach to computational creativity as *artefact-centric* creativity.

2.1 Process-Centric Creativity: Beyond the Inspiring Set

Our approach differs from standard computational creativity in a number of ways. First, we start with a computational model of people's ability to reason analogically. Our model is based on many years of focused work on the analogy process by many cognitive scientists. Our aim is not only to generate creative artefacts (analogies), but to do so in a cognitively plausible manner. We characterise this approach to computational creativity as *process-centric* creativity.

We reject any suggestion that producing creative analogies is somehow driven by an "inspiring set" of previous creative analogies. We do not wish to produce analogies that are similar to existing analogies. What we are searching for is analogies that produce the same effect, of explaining or highlight some facts. As noted Aristotle's Poetics[1] (Chapter 22) analogy is *"one thing that cannot be learnt from others"*.

A second difference with standard computational creativity is that we wish to produce an unconstrained model of creativity. We may expect our model to (re)generate a few well-known creative analogies (like Rutherford's *solar-system:atom* analogy), but we do not consider identifying these as *true* examples of creativity. That is because these are well studied analogies and they have been described in the literature in such a way as to maximise the similarity between the two domains. This has two implications. Firstly the expert's intricate knowledge is translated into a greatly simplified format, where the same relations are used to describe the source and target domains. Retrieving the simplified source can use identical token matching (used by MAC/FAC (Forbus et al., 1995)), but this would prove far less effective on the *original* problem as it was then understood. Even the semantic similarity metric used by ARCS (Thagard et al., 1990) may not provide a cue to retrieval. Secondly, the topology of the simplified domains are (generally) also identical, allowing the graph structure of the domains to play a significant role in retrieval (Plate, 1998; O'Donoghue and Crean, 2002; Gomes et al., 2003). Thus, identifying these analogies indirectly makes use of their original discovery - and does not require the same creative insight that is associated with Rutherford and others.

For example, O'Donoghue (1997) describes three successive problems that Kekule must have overcome in re-structuring the carbon-chain analogy into the famous carbon-ring analogy. Attempts were made to describe the domains in a manner more like the common understanding *before* Rutherford's famous insight - using the then dominant "plum-pudding" interpretation of the atom. However, the success of these attempts varied widely depending upon two factors. Firstly, the topological similarity of the resulting domain descriptions as the CWSG inference algorithm (Holyoak et al., 1994) generates inferences as a form of graph-completion process. Secondly, identicality or semantic similarity between the relations used to describe the source and target domains greatly influenced the likelihood of the correct analogy being drawn.

A creativity model should identify any additional creative analogies that might arise - ones that were not expected to be found. Thus, if one source domain offered some novel and useful inferences about some target domain, this *p-creative* (Boden, 1992) comparison should also be identified as creative.

A third difference with standard computational creativity arises from the fact that we do not begin with an inspiring set, as generated artefacts (the analogies) can not be evaluated by comparison with that inspiring set. Any process-centric model of creativity must be capable of differentiating between creative and non-creative artefacts, testing artefacts for the qualities associated with creativity; *novelty, quality etc* (Ritchie, 2001). However, this still leaves us with the task of assessing the goodness of the generated artefacts. Were any of the generated artefacts considered creative by humans?

Much of the remainder of this paper concerns this third point, evaluating the output of process-centric models of creativity. In the next section we examine our creative analogy model before turning our attention to analysing the analogies that were created.

[1]Aristotle made this statement about metaphor which is very similar to analogy, both being centred on a core mapping phase.

3 A Creative Analogy Machine

Wallas (1926) proposed a five-phase model of the creative process, composed of the phases: *preparation, incubation, intimation, illumination* and *verification*. This phase-model of creativity bears a striking similarity to many phase-models of the analogy process. Keane et al. (1994) identify a five-phase model of analogy composed of *representation, retrieval, mapping, validation* and *induction*. We note a particular similarity between the last three phases of Wallas' model and the central three phases of Keane's model. That is, these phases involve finding inspiration, examining the implications of that inspiration and assessing its outcome. This paper concerns the use of a computational model of analogy, consisting of the three phases of *retrieval, mapping* and *validation*[2].

As we shall see, our three-phase model is capable of finding novel analogies and of generating novel inferences. In this paper our focus is on assessing the creative potential of this multi-phase model and to do this we provide it with a knowledge-base containing a variety of domain descriptions and examine the analogies that are generated.

To test the model's creative potential we decided to generate the maximum number of analogies that can be derived from a given set of domain descriptions. Of course in a more realistic scenario, one specific domain would probably be selected as a target problem, but there were two reasons for not doing so. First there was no reason to select one domain over all others as the target problem, particularly in the absence of other facets of intelligence or domain-specific expertise. Secondly, we are attempting to explore the creative potential of a three-phase model of analogy, we are not attempting to mimic the way one specific creative analogy was discovered.

The results described in Sections 5 and 6 of this paper were generated under the following scenario. The retrieval phase selects each domain from memory in turn, treating it as a target problem. Then for each of these targets, the model retrieves every other domain in turn and treats each as a candidate source. For each resulting analogy, the inter-domain mapping is generated and the resulting inferences were generated, evaluated and recorded. So for a memory containing *n* domains, the number of analogies generated is proportional to n^2.

3.1 The Three-Phase Analogy Model

While the focus in this paper is on assessing computer generated analogies, we now provide some details on the computational model itself. In principle however, any mapping and inference models could be used. First, retrieval was a simple exhaustive process that selected each domain and passed it as a candidate source to the (current) target.

The mapping model took each source and target in turn, identifying the inter-domain mapping for that analogical comparison. Like many mapping models (Keane and Brayshaw, 1988; Keane et al., 1994; Forbus et al., 1994; O'Donoghue et al., 2006), our mapping model followed the incremental mapping approach. Before being processed, the model identified the "level" of each relation. Relations taking objects as arguments were defined as level 1, a relation taking two level n relations as arguments was defined as level n+1.

The model then identified the "root" predicates in both the source and target domains (predicates forming the root of tree structures). A "root mapping" was identified between a root in the target and a predicate at the same level in the source - with a preference that this be a root predicate. These root mappings were then elaborated to include all compatible inter-domain mappings. Our mapping model employed Gentner (1983)'s predicate identicality constraint as a preference rather than a hard restriction.

Generating inferences followed the CWSG (Holyoak et al., 1994) pattern completion algorithm. These inferences were then validated using "functionally relevant attributes" (Keane, 1985; Eskridge, 1994) that were associated with each first-order relational predicate. A simple taxonomy supported this synta-semantic process. This validation process contrasts with the more detailed but domain-specific "verification" process used by Falkenhainer (1987).

Much of this paper is devoted to analyzing the computer generated analogies that were produced by our model. Our analyses focus on the issue as to whether our model does actually generate inferences, which display the hallmarks of creativity, such as *novelty* and *quality* (Colton and Steel, 1999; Ritchie, 2001). Rather than relying on intuition, we wish to statistically assess our model by examining the artefacts it produces.

3.2 Two Collections of Test Data

In order to test our model, we need domains that the model may process. Two distinct collections of domains were used to conduct two separate computational experiments. The first collection was developed by Veale (1995) and held 14 domains, each containing from 10 to over 100 predicates. The Professions domains contained descriptions of various professions (butcher, general, politician *etc*) and though it was designed to compare models of metaphoric mapping, this use has been extended in this paper. Each domain used between 6 and 15 distinct relational predicates (ignoring duplicates). So this collection consisted of large domains described using very general relational predicates - `(depend person personal-health)` and `(depend 18-th-century-general army)`.

The second collection was developed specifically for this project and contains 81 smaller domains, ranging in size from 1 to 15 predicates. These "Sundry domains" were described by much more specific relations - `(capture army fortress)` and `(bounce golf-ball golf-green)`. This collection also contained many domains found in the analogy literature, including Rutherford's *solar-system:atom* analogy and (Duncker, 1945) *tumour:fortress* analogy. These smaller domains used an average of M=3.48 distinct relational

[2]Validation differs from verification in that it is a more broadly applicable, but less intricate means of assessing the quality of analogies and their inferences.

predicates per domain.

3.3 Initial Testing

Initial testing of our model on a few domain descriptions revealed a number of findings. First, many analogical comparisons yielded no inferences. This occurred when no appropriate inter-domain mapping could be identified and when the source domain contained no additional material to be transferred to the target.

A second finding from our initial tests revealed that almost every inference generated was a novel inference. That is, almost none of the generated inferences were identical to a predicate already contained in the knowledge base. The majority of inferences (over 99%) were formed from a novel combination of a relational predicate plus its two arguments. While we have no measure of the degree of novelty of these predicates, such a low ratio of duplicates strongly indicates that the generated artefacts should be considered novel. To remove any nonsense inferences that might be generated *eg* `(sleep idea furiously)`, the validation model classified each inference as either valid or invalid. However, we must now focus on the task of assessing the goodness of our model. Was it successful in generating creative artefacts? Did the validation model successfully remove nonsense inferences? Was validation even necessary?

4 Assessing Novel Artefacts

Computational modelling must specify the processes and representations that underlie creativity, it must also generate creative artefacts. These artefacts must thus display the qualities associated with creativity: *quality, novelty* (Ritchie, 2001), *plausibility, surprisingness, applicability, utility etc.* (Colton and Steel, 1999). The main complication in assessing these qualities arises from the fact that these artefacts are also *novel* and this novelty has some surprising implications for how the other qualities may be assessed.

Firstly, we cannot use a direct comparison between the novel artefacts and some known set of artefacts (eg the inspiring set). Thus, assessment must be conducted in other terms. Gomes et al. (2003) assess the quality of novel artefacts in terms of the quantity of identified errors in the generated artefact. Veale (2004) compares the quality of generated artefacts to an independent resources (from WordNet). Falkenhainer (1987) "verifies" analogy-based physical models in terms of how well the new model matches (or can be adapted to) other known facts and rules. In this paper we present a more general approach to the analysis of creative artefacts. Like much of cognitive science we use human evaluators to assess the goodness of the artefacts produced.

4.1 Statistical Analysis

A common methodology in cognitive science is to examine people's performance at some task. Using this evidence and other information, an hypothesis (often instantiated as a computational model) is created of their performance at that task. The goodness of the hypothesis and the model is then assessed, often using parametric statistical techniques. Among the parametric statistics used are the Pearson product-moment correlation and ANOVA (analysis of variance) tests.

However, a number of differences mean that these statistical techniques can not be used to assess computational creativity. Firstly, we are not trying to compare the performance of a set of people to the model's performance at the same task. So, assumptions about the frequency distributions that underlie many of these statistical techniques do not hold. Secondly, cognitive science assesses how well a model accounts for observed phenomena. It does not normally attempt to identify specific qualities in computer generated items.

4.1.1 Non-Parametric Statistics

To assess our model we use non-parametric statistics. Non-parametric (or distribution free) statistics make no assumptions about the frequency distribution of the variables being assessed. Thus the model's structure is not specified beforehand but is derived from the data itself. While non-parametric tests have less power than parametric tests, they are generally more robust.

While it was intended to use (human) raters to asses the goodness of the generated artefacts *post hoc*, some additional constraints were also imposed by what can be expected of raters. Newly generated items were to be evaluated independently of the domain descriptions, because presenting raters with collection of up to 100 predicates was not thought likely to be successful. Our raters did not evaluate the analogical comparisons themselves, again as rating large pairings of predicates was considered too difficult. We evaluated the analogies indirectly, based on the inferences they mandated. Again inferences were evaluated in isolation and not as collections of predicates, partly because most inferences occurred as isolated predicates. Furthermore, assessing collections of predicates would require knowing the prior context – again involving reading larg(er) collections of predicates.

In this paper we make use of two different non-parametric tests; McNemar's test and the Mann-Whitney test. Within the context of this paper, the central difference between them is that the first test compares two binary classifications, while the second test compares a binary and an ordinal classification.

4.2 McNemar's Test

In this instance we use a McNemar's test to test for the presence of an hypotheses, that our model generated artefacts displaying some of the attributes of creativity (see Hinkle et al. (1994) for details on the McNemar's test). As stated earlier, virtually all inferences were already known to be novel. So, McNemar's test was used to assess the *quality* (Ritchie, 2001) of inferences. More specifically, we assess the validity of the analogically generated inferences. (In this paper evaluation is independent of the driving analogical comparison).

More specifically, this test will allow us to test the null hypothesis, that there will be an equal number valid inferences rated-bad and invalid inferences rated-good.

We start by recording the classifications assigned to each inference by our computational model. These inferences were then given to human raters for separate assessment, so the raters were unaware of the computers classification of these items. The assigned classes are then compared to the human ratings of these materials (see Table 1). What we would like is total agreement between the assigned classifications and the human ratings.

Table 1: Confusion Matrix of Results

Assigned Class	**Human Rating +**	**Human Rating -**	**Total**
Valid +	a	b	a + b
Invalid -	c	d	c + d

In assessing these data, McNemar's test focuses on the disagreement between the categorisation and the human rating (Hinkle et al., 1994).

$$\chi^2 = \frac{(b-c)^2}{(b+c)} \tag{1}$$

McNemar's test will help us decide if our model produces valid artefacts that people think of as valid. That is, people agree that what the model categorises as a valid inference is indeed a valid inference. People also agree that the invalid inferences are invalid. Thus, the quality of generated artefacts is assessed in terms of their validity (novelty being assessed independently).

4.3 Mann-Whitney-Wilcoxon Test

To further analyse our results, a Mann-Whitney(Wilcoxon) test was also performed on the data. The Mann-Whitney-Wilcoxon test improves upon the McNemar's test by taking into account the ordinal scale used by the human raters to rate the novel artefacts. (Thus, the McNeemar's test is included in this paper for illustrative purposes). The Mann-Whitney Test is one of the most powerful of the nonparametric tests.

Mann-Whitney tests assesses if two samples come from the same distribution. The null hypothesis is that the two samples are from the same population and that their probability distributions are the same.

The two categories (valid and invalid) are combined and sorted by their rating score. The combined data are ranked and rank-sum for each category is computed (R_1 and R_2). Tied results are given the average value for that ranking group. Equation 2 details how U_1 value is calculated (an equivalent equation exists for U_2 and R_2) with U being chosen as the smaller of U_1 and U_2.

$$U_1 = mn + \frac{m(m+1)}{2} - R_1 \tag{2}$$

where m and n are the numbers of items in the two categories. For large sample sizes (n>20) an approximation can be used. Additionally, because of the presence of a large number of tied rankings among our results, we made use of a further modification to the basic formula that includes a correction factor to account for the presence of these tied rankings. Further details on the Mann-Whitney test can be found in (Siegel and Castellan, 1988).

$$\frac{z = W_x + .5 - n(N+1)/2}{[mn/N(N-1)][(N^3-N)/12 - \Sigma_{j=1}^{g}(t_j^3 - t_j)/12]} \tag{3}$$

where N=m+n, t_j is the number of tied ranks in the jth grouping, W_x is the sum of the ranks for the first category and g is the number of groupings of different tied ranks.

5 Analysis of Results

In this section we describe the results of a number of tests that were conducted on our model. A large number of analogies were generated and then assessed by examining their inferences. The quality of the resulting inferences is examined using the tests mentioned above. Additionally, some factors relating to the representation of information arise from these results, so some facets of the domain descriptions are also examined.

5.1 Experimental Set-Up

A memory was created containing all domains from two knowledge bases (described below). Each of these domains were taken in turn to serve as the target problem. Every domain was taken in turn to act as a candidate source for that target and the inter-domain mapping and inferences were generated (Holyoak et al., 1994). These inferences were then passed to a validation process, which categorised all inferences as either *valid* or *invalid.*

5.1.1 Participants and Design

Two raters were used and both raters were familiar with predicate calculus representation. All data were presented in a random order.

5.1.2 Procedure

Unrated inferences were given to human subjects who were asked to give each predicate a rating between 1 and 7. A rating of 1 represented a predicate that could not be considered credible under any circumstance, while a rating of 7 represented a predicate that could certainly be considered credible in some circumstance. A rating of 4 represented a predicate that was not obviously either credible or not credible in any circumstance.

The materials used for Experiment 1 was the inferences generated on the Professions domains. The materials used for Experiment 2 were the inferences generated on the Sundry domains. The same experimental set-up was used to produce all results.

5.2 McNeemar's Analysis

In this section we present the results of a McNeemar's analysis of the experimental data. We first present the results for the Professions domains. Next we present the results from the Sundry domains and then we compare the two results. The 7 point rating was then mapped onto a binary scale of Rated-valid or Rated-invalid, for use in the McNemar's test.

5.2.1 Experiment 1

The 14 domains from the first collection generated 196 analogies, representing each domain mapped with all other domains - including itself. The model generated a total of 175 inferences and classified 151 (86.2%) as *valid*, and 24 (13.7%) as *invalid*. Of the 175 generated inferences, 40 (approximately 1/4) were randomly selected for rating.

The average rating awarded to predicates that the model categorised as *valid* was M=2.77 (SD=1.98), while the average rating awarded to the *invalid* predicates was M=1.58 (SD=1.06). So as expected, the valid predicates were generally rated better than the invalid predicates.

Of the 20 *valid* predicates, 6 (30%) were rated as valid or potentially valid (rated >=4) by the raters, so 14 (70%) of the *valid* category were actually deemed invalid by the human raters. Of the 20 *invalid* predicates 19 (95%) were rated as invalid and 1 (5%) was rated as valid. Thus, the model appears to be better at identifying invalid predicates than it is at recognising valid predicates. This may be explained by the fact that predicates are only categorised as invalid when some specific violation of the functionally relevant attributes is identified. Otherwise, predicates are assumed to be valid.

Table 2: Assessing Generated Analogies - Collection 1

Assigned Class	**Rated Valid**	**Rated Invalid**	**Total**
Valid	3(20%)	12(80%)	15(100%)
Invalid	1(4.2%)	18(94.7%)	19(100%)

The first assessment of our computer generated items is summarised by a McNemar's test. The McNemar's test compared the classifications of the computer generated items to categorisations awarded by human raters to the same items. In this case the null hypotheses states there will be an equal number of inferences in the Invalid-RatedGood and the Valid-RatedBad conditions. The results were: #Invalid-RatedGood = 1, #Valid-RatedBad = 14, K^2=11.26 and taking α = 0.05 the null hypothesis can be rejected. p $<=$ 0.001 showing strong agreement between the two ratings, indicating that the model correctly interpreted its own output. Thus, the model was successful in generating quality artefacts that were judged to be valid by human raters.

5.2.2 Experiment 2

The second collection of 81 domains generated a total of 6561 analogies, yielding 3793 inferred predicates. Of these predicates, 2158 (56.9%) were classified as *valid* and 1635 (43.1%) inferences were categorised as *invalid* predicates.

216 valid predicates and 50 invalid predicates were randomly selected for human rating (these quantities being related to the technique which ensured a random selection was made). Of the 216 *valid* predicates, 103 (47.5%) were rated as valid or potentially valid by the raters, so 94 (43.5%) of the *valid* category were actually deemed invalid by the human raters. Of the 50 *invalid* predicates 45 (90%) were rated as invalid and 5 (10%) were rated as valid.

The average rating awarded to the *valid* predicates was M=3.47 (SD=2.35), for the *invalid* predicates was M=1.59 (SD=1.42). Thus as expected, the invalid predicates received significantly lower ratings than the valid predicates. As with Experiment 1, the invalid category is recognised with greater accuracy than the valid category.

Table 3: Assessing Generated Analogies - Collection 2

Assigned Class	**Rated Valid**	**Rated Invalid**	**Total**
Valid	94(43.5%)	122(56.5%)	216(100%)
Invalid	5(14%)	45(86%)	50(100%)

A McNemar's test was also performed to compare the model's classifications to the categorisations awarded by the raters. The results were: #Invalid-RatedGood = 5, #Valid-RatedBad = 122, K^2= 107.78, α = 0.05 so again the null hypothesis can be rejected. p $<=$ 0.0001 showing a very strong agreement between the ratings and the assigned category.

5.2.3 Discussion on Experiments 1 and 2

McNemar's test allows us to reliably reject the null hypothesis. However, a comparison between the two experiments provides greater insight.

The validation model is being very cautious about categorising relations as invalid, only doing so when there is reasonable evidence. If there is doubt about a relation's validity, it is passed as potentially valid. Thus, inferences assigned to the valid class consist of true valid inferences as well as invalid inferences on which there was insufficient information.

The average rating for the *valid* inferences in the first collection (M=2.77, SD=1.98) was significantly lower than the second (M=3.47, SD=2.35). Thus, inferences were validated less successfully on the first collection than on the second collection. However, the proportion of Valid-RatedGood = 20% in the first collection was significantly lower than on the second collection Valid-RatedGood = 43%. This can be attributed to the fact that the first collection used more general relational predicates, which are more difficult to falsify. Secondly, the second collection made greater use of relational predicates defined by functionally relevant attributes that supported the classification process.

In conclusion, it appears that the validation process is primarily responsible for the quality of inferences in the valid and invalid categories. Domains that are described using more specific relations (from lower-down a taxonomy) allow the validation process to operate more accurately.

5.3 Mann-Whitney(Wilcoxon) Analysis

A Mann-Whitney analysis waas conducted on our results. As stated earlier, the main results below counter for the presence of large number of tied results. The presence of tied rankings was a greater factor in the analogies generated from the second collection than on the first collection of domains.

We first present the results for the Professions domains and then results from the Sundry domains before comparing the two results.

The materials and method used in this experiment were the same as in the previous analysis.

The null hypothesis tested in this section is that the two samples are from the same population and that their probability distributions are the same.

5.3.1 Experiment 3

With the formula given above, the results for our Mann-Whitney test are : R_1 323.5, R_2 498, U = 112, z=2.3 (p_1 <0.0107, p_2 <0.0214).

However, when we use the Mann-Whitney test that is adjusted for the presence of many tied results. z=2.49, (p_1 <0.0064, p_2 <0.02).

This result allow us reject the null hypothesis, that the valid and invalid categories are drawn from the same population. We can thus adopt the alternate hypothesis that the mean of the valid category is greater than the invalid category. Thus our analogy model is indeed generating quality analogical inferences.

5.3.2 Experiment 4

With the formula given above, the results for our Mann-Whitney test are : R_1 30057.5, R_2 5414, U = 4147.5, z=2.55 (p_1 <0.005, p_2 <0.0108).

However, if we include the correction factor to account for the presence of tide results in our ranking, then z = 5.92 (p_1 <0.0001, p_2 <0.0001).

Thus we reject the null hypothesis in favour of the alternate hypothesis, which is that the median of the valid inferences is greater than the median of if invalid inferences. Alternatively we may state that the valid inferences have stochastically greater ratings than the invalid inferences.

Again this was the hoped for result and shows that our analogy model does generate quality inferences.

5.3.3 Discussion on Experiments 3 and 4

As expected, these results indicate that the null hypothesis can be rejected. Again, the results from the Assorted collection given graeter confidence for this conclusion than do the Professions results.

5.4 Known Creative Analogies

For clarity, we shall report separately on the accuracy of our model to re-generate known creative analogies. As discussed in Section 2.1 we do not consider the following results to be examples of true creativity - because the domain descriptions do not accurately reflect the problems that were creatively solved in each instance. However, they do provide some positive evidence for the creative potential of the analogy model.

Because the creative analogies were known *a priori*, we do not need to use McNemar's test. The model generated and validated the correct inferences for 7 of 10 (70%) (known) creative analogies and thus was quite successful in (re)generating these analogies.

While no new creative analogies were discovered on these knowledge-bases, we believe that creative analogies could be discovered by our model. These results show us that creative analogies occur exceptionally rarely. A challenge for the future is to acquire more domain descriptions to see if any creative analogies are generated. A related challenge is to improve the evaluation process in order to focus on the more promising and creative analogical comparisons.

6 Conclusion

The traditional approach to computational creativity attempts to generate new items belonging to an "inspiring set". But this inspiring set also plays a role in evaluating the creativity model.

We describe an alternative *process-centric* approach to computational creativity that does not utilise an inspring set. Thus, evaluating these models must rely on alternative methods. This paper describes two nonparametric statistics techniques, namely McNeemar's test and the Mann-Whitney test. These tests evaluate artefacts that have been rated on binary and ordinal scales respectively.

These statistical techniques were used to evaluate a model of analogical reasoning that has been adapted to operate as an analogy generating machine. This model encompassed the three core phases in the analogy process, namely: *retrieval, mapping* and *validation.* The model was used on a knowledge base containing two distinct collections of domains, to assess its performance and see if any novel or creative analogies might be generated. Every domain was used as a target in conjunction with each other candidate source domain. This generated the maximum number of analogical inferences allowing us to test the creative potential of our model.

The resulting inferences were evaluated by the model itself, selecting inferences of greater quality. These inferences were recorded and given to human raters who assessed the accuracy of the analogy production system. A McNemar's test was used to compare and assess the automatically assigned classification against human ratings of the same artefacts. This illustrated that the model was successful in generating quality inferences.

Known examples of creative analogies were identified as expected (such as Rutherford's *solar-system:atom* analogy). However, such examples are not considered as *truly* creative as they have been described in such a way as to maximise the similarity between the two domains making their re-discovery almost inevitable. No *truly* creative analogies were identified among the subset of inferences rated by the raters.

Interesting differences between the two collections produced differences in the generated results. The collection using more general (or abstract) relational-predicates made generating the mapping easier, but made validation less accurate. In contrast, the collection using more specific relational-predicates made identifying the inter-domain mapping more difficult, but allowed more accurate validation of inferences.

As far as we know, this is the first work towards automatically generating analogies. While the analogies reported in this paper were not found to be creative, we believe a larger knowledege-base will provide more fruitful results. More accurate and complete models of each phase of analogy may help improve the quality of results produced by the model. Modifying the model's parameters may even produce a model with a greater creative capacity than human analogisers.

Acknowledgements

My sincere thanks to the anonymous reviewers of IJWCC 2007 (one in particular) for helpful comments and suggestions on this paper.

References

Boden, M. A. (1992). *The Creative Mind*. Abacus.

Brown, T. L. (2003). *Making Truth: Metaphor in Science*. University of Illinois Press, New York, USA.

Colton, S. and Steel, G. (1999). Artificial intelligence and scientific creativity. *Artificial Intelligence and the Study of Behaviour Quarterly*, 102.

Curie, M. (1923). *Pierre Curie*. Macmillan.

Dunbar, K. (2001). *The Analogical Mind*, chapter The Analogical Paradox: Why Analogy is so Easy in Naturalistic Settings, Yet so Difficult in the Psychological Laboratory.

Dunbar, K. and Blanchette, I. (2001). The in vivo/in vitro approach to cognition: The case of analogy. *Trends in Cognitive Sciences*, 5(8):334–339.

Duncker, K. (1945). On problem solving. *Psychological Monographs*, 5:whole no. 270.

Eskridge, T. C. (1994). *Analogy, Metaphor and Reminding*, chapter A Hybrid Model of Continuous Analogical Reasoning. Ablex, Norwood, NJ.

Eysenck, M. W. and Keane, M. T. (1995). *Cognitive Psychology*. Taylor Francis, Erlbaum, UK.

Falkenhainer, B. (1987). An examination of the third stage of the analogy process: Verification-based analogical learning. In *Proc. IJCAI*, pages 260–263.

Falkenhainer, B., Forbus, K., and Gentner, D. (1989). The structure mapping engine: Algorithm and examples. *Artificial Intelligence*, 41:1–63.

Forbus, K., Ferguson, R., and Gentner, D. (1994). Incremental structure-mapping. In *Proc. 16th Cognitive Science Society*, pages 313–318.

Forbus, K., Gentner, D., and K., L. (1995). Mac/fac: A model of similarity-based retrieval. *Cognitive Science*, 19:141–205.

French, R. M. (2002). The comptational modeling of analogy-making. *Trends in Cognitive Sciences*, 6(5):200–205.

Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7:155–170.

Gomes, P., Seco, N., Pereira, F. C., Paiva, P., Carreiro, P., Ferreira, J. L., and Bento, C. (2003). The importance of retrieval in creative design analogies. In *Proc. IJCAI 3rd Workshop on Creative Systems*, Acapulco, Mexico.

Hinkle, D. E., Wierrsma, W., and Jurs, S. G. (1994). *Applied Statistics for the Behavioral Sciences*. Houghton Mifflin, Boston, USA.

Hoffman, R. (1995). Monster analogies. *AI-Magazine*, 3:11–35.

Holyoak, K., Novick, L., and E., M. (1994). *Analogy, Metaphor and Reminding*, chapter Component Processes in Analogical Transfer, pages 113–180. Ablex Norwoord, N.J., USA.

Keane, M., Ledgeway, T., and Duff, S. (1994). Constraints on analogical mapping: A comparison of three models. *Cognitive Science*, 18:387–438.

Keane, M. T. (1985). On drawing analogies when solving problem:. *British Journal of Psychology*, 76:449–458.

Keane, M. T. and Brayshaw, M. (1988). *Third European Working Session on Machine Learning*, chapter Indirect Analogical Mapping. Pitman, London, UK.

Koestler, A. (1964). *The Art of Creation*, volume 1. Picador, London.

O'Donoghue, D. (1997). Towards a computational model of creative reasoning. In *Conference on Computational Models of Creative Cognition (CMOCC)*, Dublin City University, Ireland.

O'Donoghue, D. and Crean, B. (2002). Retrieving creative analogies. In *ECAI - Workshop on Creative Systems*, pages 56–66, Lyon, France.

O'Donoghue, D. P., Bohan, A., and Keane, M. (2006). Seeing things : Inventive reasoning with geometric analogies and topographic maps. *New Generation Computing*, 24:267–288.

Plate, T. (1998). Structured operations with distributed vector representations. In *Advances in Analogy Research*, Sofia, Bulgaria,.

Ritchie, G. (2001). Assessing creativity. In *Proc. AISB Symposium on AI and Creativity*, pages 3–11, York, England.

Siegel, S. and Castellan, J. N. (1988). *Nonparametric Statistics*. McGraw-Hill.

Thagard, P., Holyoak, K. J., Nelson, G., and Gochfeld, D. (1990). Analogue retrieval by constraint satisfaction. *Artificial Intelligence*, 46:259–310.

Veale, T. (1995). *Metaphor, Memory and Meaning*. Ph.d. diss., Trinity College, Dublin.

Veale, T. (2004). Pathways to creativity in lexical ontology. In *Proc. 2nd Global WordNet Conference*.

Wallas, G. (1926). *The Art of Thought*, volume 1. Cape, London.

A GENERATIVE GRAMMAR FOR PRE-COLUMBIAN ARTISTIC PRODUCTION: THE CASE OF EL TAJÍN STYLE

Manuel Álvarez Cos
U.H. Loma Hermosa 73c- 406,
Col. Loma Hermosa, c.p.11200,
México, D.F, México
arqueomac@yahoo.com

Rafael Pérez y Pérez
Departamento de
Tecnologías de la Información,
Universidad Autónoma
Metropolitana (UAM-Cuajimalpa)
Avenida Constituyentes 1054,
Col. Lomas Altas, Miguel Hidalgo,
México, D. F., C.P. 11950
rperez@correo.cua.uam.mx

Atocha Aliseda Llera
Instituto de
Investigaciones Filosóficas,
Universidad Nacional
Autónoma de México (UNAM)
Circuito Mario de la Cueva, s/n.
Ciudad Universitaria México,
c.p. 04510, México, D.F., México
atocha@servidor.unam.mx

Abstract

This paper presents a shape grammar for the production and recognition of images conformed to the artistic style of El Tajín, an archaeological city in México. After an introductory review of previous archaeological and computational works, the paper describes the syntax and the process by which the Generative Grammar of El Tajín Style (GGTS) composes depictions of images. Subsequently, the paper presents a brief description of the Prolog implementation of GGTS and brings forward some reasons for considering GGTS an anthropological cognitive model of creativity. The paper finishes discussing the virtues and shortcomings of the Generative Grammar of the El Tajín Style (GGTS).

Keywords: Computational creativity, shape grammar, cognitive anthropology.

1 Introduction

Most computational applications for archaeology focus on statistical analysis, geographical information systems, automated cartography, virtual reality, 3D scanning, and internet applications (Wise and Miller, 1997; Prinke, 2005; Dawson and Richard, 2005; Bellanger et al., 2006; Ebert, 2006). However, automatic tools for the description, classification, interpretation and generation of iconographic material are necessary too. An overview of the methods currently used in iconographic studies is done by Camiz (2004). He describes some notational methods, and discusses their conveniences and limitations. All those methods are algebraic languages where symbols representing spatial relations articulate symbols standing for icons (e.g., {*iconName1 . iconName2*}, where "." may mean "is besides"). Unfortunately, none of these languages can be considered an objective definition, because none of them generates novel instances of a given style.

In contrast, shape grammars have been more successful in this respect. The first examples were designed in the early 70s, but it was in the 80s when grammars were used to define specific artistic styles (Kirsch and Kirsch, 1988), and in the 90s when they were combined with pattern recognisers (Scidmore and Bajcsy, 1979; Bajcsy et al., 1984; Matsello et al., 2002) [1].

In this paper we present a Generative Grammar similar to those created by Kirsch and Kirsch. The grammar, referred to as the **G**enerative **G**rammar of the El **T**ajn **S**tyle (GGTS), does not work with pixel maps. Instead, it generates and recognises surrogate representations.

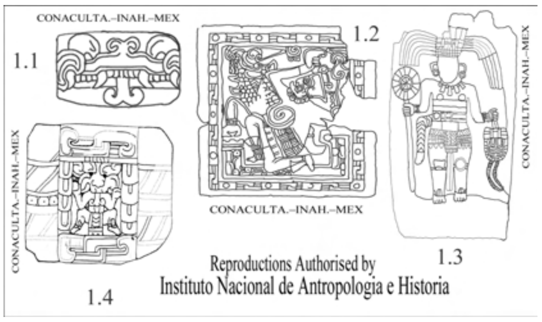


Figure 1: Examples of the El Tajín Style

2 Former Archaeological Studies

El Tajín City is an archaeological site located at the East Coast of México (300a.d-900a.d). The reliefs of the city have been studied in multiple occasions, although the bulk of these works concentrates in the symbolic or religious content of the sculptures. Amidst the few works studying the compositional structure of the images, the most cited are those authored by Proskouriakoff (1953), Kampen (1972), Castillo-Peña (1995), Pascual Soto (1990) and Pascual Soto (2005).

Among the numerous and decisive contributions made by all these researchers, we review the one of Pascual Soto (1990), for only this author employs a formal language as part of his proposal. His syntax, very similar to those discussed by Camiz (2004), describes a compositional structure in which a "base" (principal) symbol is surrounded by outer symbols. This syntax has two shortcomings. It does not allow any kind of inference and it does not offer

[1]For a brief history of shape grammars consult http://www.shapegrammar.org/

any objective method for the translation of a formula into the image it depicts. The formulas indicate that an icon (complex or simple) is over, below, at the left, or at the right of a central icon. They do not give any hint about the exact position and appearance of the symbols[2].

To sort out this problematic, it was sought a formal representation that could model the cognitive skill behind the design and contemplation of the sculptures. This representation was constructed with an inductive method. First, a sample of the known sculptures was analysed in a hierarchical manner in order to detect the compositional regularities characterising the El Tajín Style. Later, these regularities were extrapolated and represented with formulas expressing the common layout of similar scenes and objects. At the end, since these formulas depicted similitudes of different levels of abstraction, it was necessary to define rules describing which compositional arrangements could be jointed with what others. These rules, once organised, gave rise to a generative grammar.

3 Compositional Principles of EL Tajín Style

Every relieve in El Tajín City is engraved over a stele or a pile of stones. The engraved images show single objects or scenes (characters inside a context). Scenes are always framed, while single objects may appear without a frame. Single objects are pumpkins, serpent heads (Fig. 1.1), and solitary anthropomorphic figures (Figs. 1.3, 1.4). Scenes, on the other hand, represent human sacrifices, offering rituals, processions of warriors with prisoners, sat rulers guarded by floating serpents, and anthropomorphic figures kept in architectural structures (Fig. 1,2).

Each of those themes is repeatedly observed, although in each occasion slight variations are introduced in the elements conforming the image. This *exploratory* creativity constrained by rigid conventions, common in most forms of religious art, is the one modeled by GGTS.

4 Formal Structure of GGTS

GGTS comprises an **I**nferential mechanism (I) for the generation of relieves, a list (*P*) of 1267 **P**roductions hierarchically arranged, a start symbol (Sculp), a list (*N*) of 1125 Non-terminal symbols, a dictionary (*T*) of 2140 Terminal symbols, a set (*KV*) for the storage of **K**nown **V**aluable images, and a set (*KR*) for the storage of **K**nown **R**ejected images $(GGTS = (I, P, Sculp, T, N, KV, KR))$.

[2]Pascual Soto (1990: p208) describes the relieve in Figure 2 with the formula: *'(97.)(.?)(?:) /(1004 +, 414)/ (.)(.97)'*. This formula says that the *base* symbol (/+/) is the *association* (,) of the symbols *1004* and *414*; that *over* (:) the *base* symbol there's an *unclassifed* symbol (?); that symbol *97* is to the *right* (.97) of an *unknown* symbol located *at the right* (.) of the *base* symbol; that symbol *97* is to the *left* (97.) of an *unclassified* symbol located at the *left* of the *base* symbol.

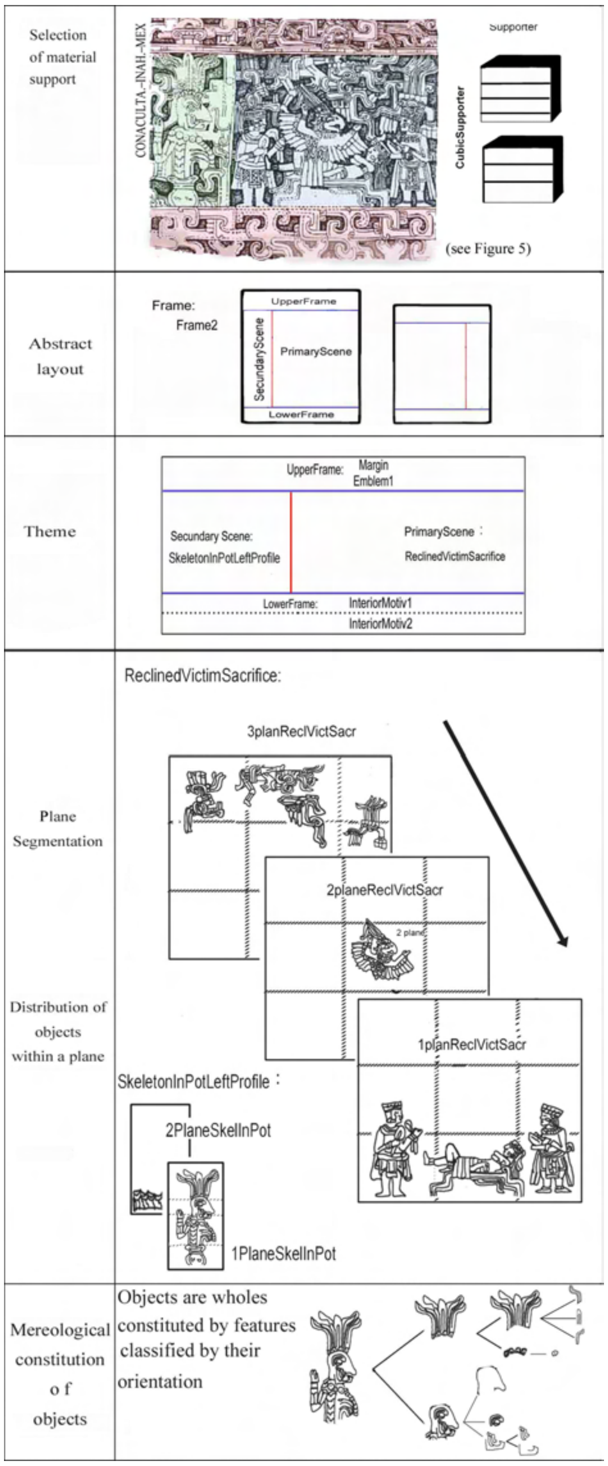

Selection of material support
CONACULTA–INAH–MEX
Supporter
CubicSupporter
(see Figure 5)
Abstract layout
Frame:
Frame2
UpperFrame
SecundaryScene
PrimaryScene
LowerFrame
Theme
UpperFrame: Margin Emblem1
Secundary Scene: SkeletonInPotLeftProfile
PrimaryScene : ReclinedVictimSacrifice
LowerFrame: InteriorMotiv1 InteriorMotiv2
Plane Segmentation
Distribution of objects within a plane
ReclinedVictimSacrifice:
3planReclVictSacr
2planeReclVictSacr
2 plane
1planReclVictSacr
SkeletonInPotLeftProfile :
2PlaneSkelInPot
1PlaneSkelInPot
Mereological constitution of objects
Objects are wholes constituted by features classified by their orientation

Figure 2: Levels of composition within a relieve

I is a non-monotonic system based on *Closed World Assumption* $(CWA\,(S) = ((S \nvdash P,) \rightarrow (S \mid\!\sim \neg P)))$ and *Modus Ponens* $((P, P \rightarrow Q) \vdash Q)$.

The productions in *P* have the form:

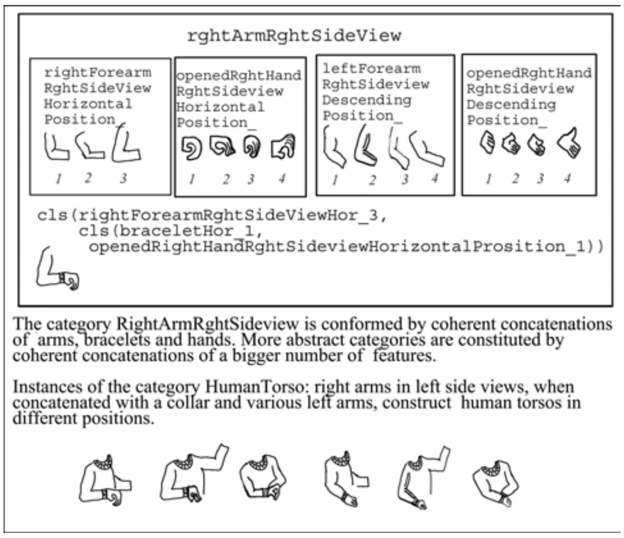
$$IconName \rightarrow$$
$$Preposition < IconName//IconName >.$$

Here, *Preposition*, as in many natural language grammars, refers to a symbol that specifies the spatial relation that articulates two objects, in this case icon names. There are non-terminal and terminal prepositional symbols.

Table 1: Prepositional symbols. Every prepositional relation has a terminal symbol (`cls_1`) and a non-terminal symbol (`cls`)

Looks like this	Is described by this	Is formalized by this
□ ○	□ is contiguous to ○	C(□, ○)
⌇▭	⌇ contacts ▭ from below	CB (⌇, ▭)
□—	□ contacts left extreme of ——	CLE (□, —)
○▯	○ contacts left side ▯	CLS(○, ▯)
○▭	○ contacts top of ▭	CT (○, ▭)
∩∪	the nodes of ∩ contact the nodes of ∪	CN(∩ , ∪)
⌓○	the concavity of ⌓ holds ○	CV(⌓,○) / CV(◡,○)
■■	■ is behind of ▨	B (■,▨)
▭△	▭ encloses △	ENC(▭, △)
▱△	— presents △	P(— , △)
▱—	▱ supports —	S(▱, —)
▭△	▭ is over △	O(▭, △)

The angle brackets represent the punctuation marks that limit the scope of a prepositional symbol, and the double slash represents the marks that separates the names of the two elements participating in a topological relation. Both kinds of marks have a non-terminal and terminal version. Non-terminal parentheses are"[" and "]", and the terminal ones are "(" and ")"; whereas non-terminal separatrix is "/", and terminal separatrix is ",".


rghtArmRghtSideView
rightForearm RghtSideView Horizontal Position_
1 2 3
openedRghtHand RghtSideview Horizontal Position_
1 2 3 4
leftForearm RghtSideview Descending Position_
1 2 3 4
openedRghtHand RghtSideview Descending Position_
1 2 3 4

```
cls(rightForearmRghtSideViewHor_3,
    cls(braceletHor_1,
     openedRightHandRghtSideviewHorizontalProsition_1))
```

The category RightArmRghtSideview is conformed by coherent concatenations of arms, bracelets and hands. More abstract categories are constituted by coherent concatenations of a bigger number of features.

Instances of the category HumanTorso: right arms in left side views, when concatenated with a collar and various left arms, construct human torsos in different positions.

Figure 3: Examples of feature categories

IconName symbols may refer to terminal icon names (names of a basic features), or to non-terminal icon names, where a non-terminal icon name is that which can be transformed into a terminal icon name or into a chain of non-terminal symbols of the form $Preposition < iconName//iconName >$.

The terminal and non-terminal icon names are abbreviated phrases describing the *asymmetric geometry* of an icon category or icon feature. This is, the terminal and non-terminal icon names of the grammar describe the orientation of an object, as is perceived from a vantage point (Fig. 3).

The two types of icon names are distinguished by a typographical mark. Feature names end with the suffix "_*n*", where *n* could be any natural number.

As an example, take the case of the symbol `rghtArmRghtSideview`; it means *right arm seen from its right side view*. But given that it has no suffix, then it is a non-terminal icon name. Hence, it is not possible to associate a specific drawing to `rghtArmRghtSideview`. In order to be instantiated, it must be transformed into a topological relation sustained by two or more icon names.

The abstractness of a non-terminal icon name is represented with groups of productions with equal symbols on the left side of the arrow, but different chains of symbols on the right side of the arrow. Consider the following examples:

(1)

`rghtArmRghtSideview` $\rightarrow$
`cls[rghtForearmRghtSideviewDescend/`
`cls[braceletDescend/`
`opnRghtHndRghtSideviewDescend]].`

(2)

`rghtArmRghtSideview` $\rightarrow$
`cls [rghtForearmRghtSideviewHorizontal/`
`cls[braceletHor/`
`opnRghtHndRghtSideviewHoriz]].`

These productions express that the category `rghtArmRghtSideview` includes two subcategories. Thus, every time a *right arm seen from its right side view* has to be constructed, it might end up with the appearance of a:

> *(1) Right forearm seen from its right side view in descending position, contacting the left side* (CLS) *of a bracelet in descending position*, which *contacts the left side of an open right hand seen from its right side view in descending position*.

Or it might look like a:

> *(2) Right forearm seen from its right side view in descending position, contacting the left side* of a *bracelet in horizontal position, that contacts the left side* of an *open right hand seen from its right side view in horizontal position*.

Given this set up, it is possible to describe GGTS as a hierarchical collection of compositional conventions. In this hierarchy, the most abstract formulas associate a material support and a surface with a general scene distribution. Less abstract formulas define which icon types are to be located in which positions. And the most concrete formulas describe the way in which observable features (terminal icon names) conform a coherent and valuable image (Figs.2 and 3).

5 Derivation of Relieves with GGTS

The constitution of an image is a compositional process. And it is precisely this fact what allows both to recognise known instances of the El Tajín Style as well as to create new ones.

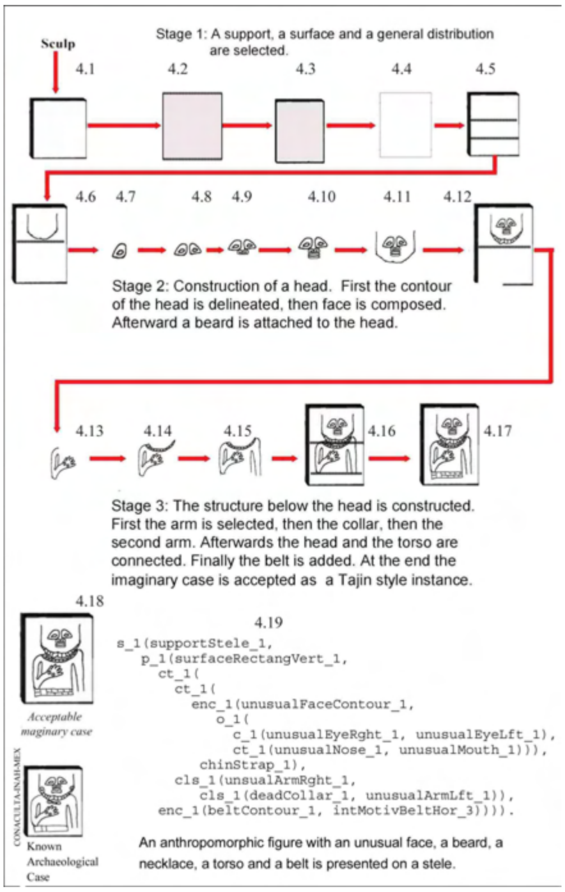
5.1 Production of Novel Images

The production of a novelty always begins with the introduction of the start symbol (Fig.4.1), and ends up with the construction of a formula describing the position of every basic feature conforming the image (Fig. 4.19). In Figure 4 (Fig.4.2- 4.4), after its introduction the start symbol (`Sculp`) is substituted by a formula that articulates a support (`stele`) with a rectangular surface and an unframed layout (`board1`). Afterwards, a drawing plan is decided by picking out a general theme: `unusualAnthropomorphicFigure`. This last decision causes the image to be segmented into three sections (Fig 4.5), one over the other: `unusualHead`, `unusualTorso`, and `belt`. Subsequently, `unusualHead` is expanded into, or rewritten as, the main sections of a head: an `unusualBeard` below an `unusualFaceCountour` that `encloses` an `unusualFace1`. Then `unusualFaceContour` is transformed into `unusualFaceCountour_1`(Fig 4.6). Afterwards, `unusualFace1` is subdivided into four sections: `leftUnusualEye`, `rightUnusualEye`, `unusualNose`, and `unusualMouth`: the first next to the second, the third below the former two and above the fourth one. Subsequently (Figs. 4.7- 4.11), the four subsections are filled out with the terminal icon names `leftUnusualEye_1`, `rightUnusuaEye_1`, `unusualNose_1`, and `unusualMouth_1`. Once this is done, the symbol `unusualBeard` is transformed into `unusualBeard_1` (4.12). At this moment the head has been finished.

The torso is constructed likewise. The symbol `unusualTorso` is transformed into a topological relation sustained by three non-terminal icon names: a *left arm*, a *necklace* and a *right arm*. Afterwards, a torso is instantiated by transforming each of those non-terminal icon names into suitable terminal icon names (Fig. 4.13- 4.16). Finally, the non-terminal icon name `belt` is substituted by a *decoration* inside the *contour of a belt* (Fig. 4.17).

Given that the image has been constructed by strict fulfilment of accepted rules, the value of the image is granted. It is a well formed object. Nevertheless, its novelty still has to be evaluated. This is done by searching for a copy of the constructed image in *KV*, which is nothing more than a list of previously constructed descriptions. If there is no such copy, then the image will be taken as a novelty and will be included in *KV* (this process is explained in section 5.2).

This criterion may be considered a naive instance of what may be called creative. However, the approach here presented is tied to create new things out of a language, and also tied to comply with a set of internal rules. In any case, an image thus created is totally new with respect to what ever was known before, and this is why we call it a novelty. Withal, the archaeological evidence found at El Tajín seems to show that, at least in certain societies, a novel and valuable artistic piece can be obtained by an *exploratory strategy* interested in very slight variations.


Stage 1: A support, a surface and a general distribution are selected.
Sculp
4.1
4.2
4.3
4.4
4.5
4.6
4.7
4.8
4.9
4.10
4.11
4.12
Stage 2: Construction of a head. First the contour of the head is delineated, then face is composed. Afterward a beard is attached to the head.
4.13
4.14
4.15
4.16
4.17
Stage 3: The structure below the head is constructed. First the arm is selected, then the collar, then the second arm. Afterwards the head and the torso are connected. Finally the belt is added. At the end the imaginary case is accepted as a Tajin style instance.
4.18
Acceptable maginary case
CONACULTA-INAH-MEX
Known Archaeological Case
4.19
An anthropomorphic figure with an unusual face, a beard, a necklace, a torso and a belt is presented on a stele.

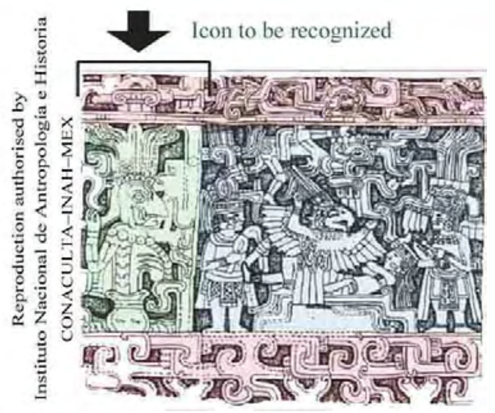
```
s_1(supportStele_1,
   p_1(surfaceRectangVert_1,
     ct_1(
       ct_1(
         enc_1(unusualFaceContour_1,
            o_1(
              c_1(unusualEyeRght_1, unusualEyeLft_1),
              ct_1(unusualNose_1, unusualMouth_1))),
          chinStrap_1),
       cls_1(unsualArmRght_1,
          cls_1(deadCollar_1, unusualArmLft_1)),
     enc_1(beltContour_1, intMotivBeltHor_3)))).
```

Figure 4: Construction of a new image

5.2 Recognition of Novelties

The recognition process is a compositional sequence too. As shown in the example of Figure 5, the process starts with the reception of an input formula describing an image. This initial input is taken as a “what is this?” question. This sets out the production of a hypothesis about its general layout (Figs. 6.1, 7.2). But, given that the hypothesis is a non-terminal icon name (`s[quadrangularSupport...]`), it cannot be contrasted with the input formula, which is entirely composed of terminal symbols. Thus, the non-terminal icon name is recursively transformed until the structure of a frame is selected(Figs. 6.2-6.4, 7.2-7.5).

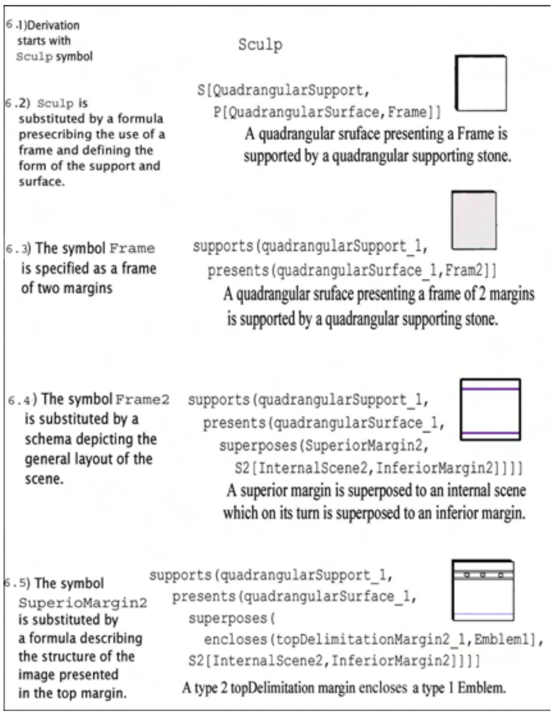
Later on (Figs. 6.5, 7.6), , `SuperiorMargin2` is transformed into a `TopDelimitationMargin2` that encloses an `Emblem1`. Then `TopDelimitationMargin2` is transformed into `topDelimitationMargin2_1`, which is, as its typography attests, a terminal icon name that can be matched with the input formula .


Icon to be recognized
Reproduction authorised by
Instituto Nacional de Antropologia e Historia
CONACULTA–INAH–MEX

```
s_1(quadrangularSupport_1,
p_1(quadrangularSurface_1,
s_1(
enc_1(topDelimitatioMargin2_1,
c_1(
cls_1(rearDecoAbstIconHorRghtSideviewUp_8,
cls_1(ct_1(
skelEyebrowRghtSideview_7, jawAbstIconFront_1),
ct_1(leafLftSideviewHorUp_1,
supDentitionRghtSideviewHorUp_6))),
cls_1(cls_1(
cls_1(supvoluteBagRght_4, InfVoluteBagRght_1),
...)...)..)..))))
```

Figure 5: Reclined Victim Sacrifice and Input formula depicting the highlighted fragment

After a successful first matching (Fig.7.6), the compositional process continues until the icon described in Figure 5 is accepted as a valuable El Tajín style image. Finally, the novelty of the image is confirmed by checking that there is no record of it in *KV* (7.16).



6.1)Derivation starts with `Sculp` symbol

```
Sculp
```

6.2) `Sculp` is substituted by a formula presecribing the use of a frame and defining the form of the support and surface.

```
S[QuadrangularSupport,
  P[QuadrangularSurface,Frame]]
```

A quadrangular sruface presenting a Frame is supported by a quadrangular supporting stone.

6.3) The symbol `Frame` is specified as a frame of two margins

```
supports(quadrangularSupport_1,
  presents(quadrangularSurface_1,Fram2]]
```

A quadrangular sruface presenting a frame of 2 margins is supported by a quadrangular supporting stone.

6.4) The symbol `Frame2` is substituted by a schema depicting the general layout of the scene.

```
supports(quadrangularSupport_1,
  presents(quadrangularSurface_1,
    superposes(SuperiorMargin2,
      S2[InternalScene2,InferiorMargin2]]]]
```

A superior margin is superposed to an internal scene which on its turn is superposed to an inferior margin.

6.5) The symbol `SuperioMargin2` is substituted by a formula describing the structure of the image presented in the top margin.

```
supports(quadrangularSupport_1,
  presents(quadrangularSurface_1,
    superposes(
      encloses(topDelimitationMargin2_1,Emblem1],
    S2[InternalScene2,InferiorMargin2]]]]
```

A type 2 topDelimitation margin encloses a type 1 Emblem.

Figure 6: Initial steps of a derivation sequence

An alternative scenario would be that in which no successful hypotheses could be derived from the productions of the grammar. In this case, the absence of an applicable production would enable the storage of the formula into the set *KR*, although this conjecture could be retracted after the acquisition of new knowledge. In this last case, it would suffice to search in *KR* for formulas that could be recognised as valuable, despite its previous rejection.

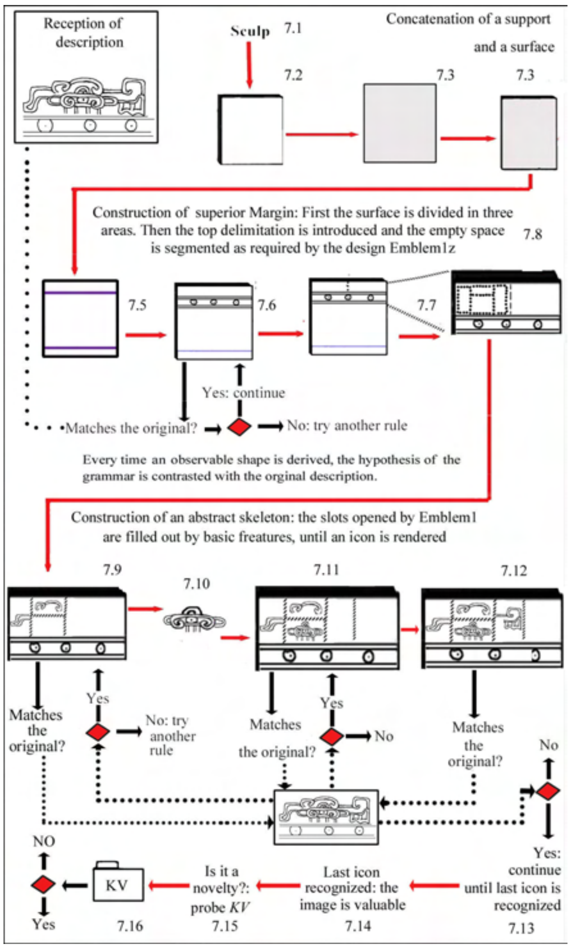

Reception of description
Sculp 7.1
7.2
Concatenation of a support and a surface
7.3
7.3
Construction of superior Margin: First the surface is divided in three areas. Then the top delimitation is introduced and the empty space is segmented as required by the design Emblem1z
7.8
7.5
7.6
7.7
Yes: continue
Matches the original?
No: try another rule
Every time an observable shape is derived, the hypothesis of the grammar is contrasted with the orginal description.
Construction of an abstract skeleton: the slots opened by Emblem1 are filled out by basic freatures, until an icon is rendered
7.9
7.10
7.11
7.12
Matches the original?
Yes
No: try another rule
Matches the original?
Yes
No
Matches the original?
No
NO
Yes: continue until last icon is recognized
7.13
Last icon recognized: the image is valuable
7.14
Is it a novelty?: probe KV
7.15
KV
7.16
Yes

Figure 7: Recognition process

6 Prolog implementation

The Prolog version of the grammar (GGTS.pl) runs as a *depth-first search* that traverse a searching space from left to right. This basic mechanism always starts a search in the same point (`Sculp`) and may follow an exhaustive route. However, if the user introduces a query specifying a type of theme or the exact position of a feature or chain of features, the search is done differently. In these two cases, GGTS.pl starts in the same initial node and derives part of an image, but if the created segment does not suit the conditions stated by the user, the image is cancelled and a whole category of similar images is discarded, thus preventing the exploration of certain regions of the searching space.

A second manner in which the searching process is hastened is through a modular structure. GGTS' productions are sorted in accordance to their level of abstraction and general subject. This set up lessens the number of lines to be read when searching for a rule.

The productions of GGTS.pl have one of two forms:

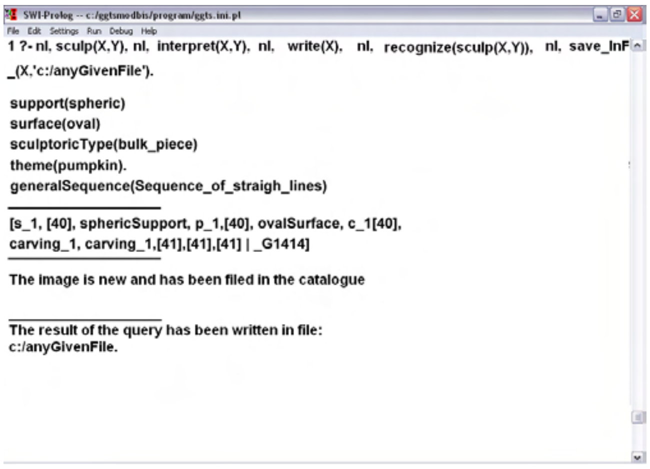
(a)

nonTerminalIconName(A,F):-
Preposition (A,B),
OpeningParenthesis (B,C),
NonTerminalIconName(C,D),
NonTerminalIconName(D,E),
ClosingParenthesis(E,F),
ParsingProcess.

(b)

nonTerminalSymbol([terminalSymbol | Tail],Tail).

In (a), the lines ending with two variables inside a parenthesis are list constructors. They introduce terminal icon names into specific positions of a list, enabling the transformation of the `Sculp` symbol. The line "*parsingProcess*" is the call for a procedure that registers the successful use of the production. On the other hand, the form (b) is used to define a terminal symbol. In this way, at the end of a successful search, the list constructors compose a chain of basic features, while the registration process gives an interpretation of that chain.


SWI-Prolog -- c:/ggtsmodbis/program/ggts.ini.pl
File Edit Settings Run Debug Help

```
1 ?- nl, sculp(X,Y), nl, interpret(X,Y), nl, write(X), nl, recognize(sculp(X,Y)), nl, save_InF
_(X,'c:/anyGivenFile').

 support(spheric)
 surface(oval)
 sculptoricType(bulk_piece)
 theme(pumpkin).
 generalSequence(Sequence_of_straigh_lines)
________________
[s_1, [40], sphericSupport, p_1,[40], ovalSurface, c_1[40],
carving_1, carving_1,[41],[41],[41] | _G1414]
________________
The image is new and has been filed in the catalogue

________________
The result of the query has been written in file:
c:/anyGivenFile.
```

Figure 8: Example of a query

Besides the productions of the grammar, there is a set of seven predicates that allow the user to 1)print or save the formulas generated during a session, 2)run test suites, 3)sort formulas into the sets *KV* and *KR*, 4)decide if a given formula is valuable and new, 5)revise the set *KR* (retract conjectures), and 6)obtain an interpretation for a given formula. Figure 8 illustrates some of these predicates. The query tells GGTS.pl to construct and describe a sculpture, intepret it, save it in KV (if possible), and store a copy in "*a given file*". The result of each command is separated by horizontal lines.

The accuracy of GGTS.pl has been partially contrasted against the official catalogue of the Mexican National Institute of Anthropology (Castillo-Peña, 1995). The productions now registered in the program are reliable. They accept what they are supposed to, and reject paradigmatic examples of the styles created at other related cities (Teotihuacan, Tula and Tecnochtitlan). On the other hand, the grammar is still an incomplete model and the rules are somewhat rigid. Nevertheless, given the non-monotonic structure of the program, when the missing compositional conventions of El Tajín style get to be included, it will be possible to update the sets *T, N* and *P*, revise the set *KR*, and expand the set *KV* (section 4).

7 GGTS as an Anthropological and Cognitive Model of Creativity

Cognitive archaeology studies the human ability to construct and use symbols (Renfrew et al., 1994; Preucel, 2006). Its principal interests are design, representation (production and utilisation of iconic embodiments of reality), planning, measurement, religion and symbolic social control (preservation of social alliances). In accordance with these objectives, it has been shown that GGTS.pl has access to a set of possible symbols quite similar to the one accessed by the inhabitants of El Tajín city. But for the sake of anthropological adequacy, it is also of interest to see if the functional sequence of the compositional process done by the grammar has any similitude with the functional sequence of the human brain.

Psychological experiments interested in eye movements have shown that visual scenes are scanned in a piecemeal manner, following the guidance of mental schemata that predict the general layout of a scene (Henderson and Ferrerira, 2004; Cowan and Moorey, 2006). Likewise, it seems to be the case that the same mental schemata are used to guide bodily movements, such as the motion of a sculptor's arm (Mushiake et al., 1999; Carmena and Nicoleli, 2003). In this respect, the step-by-step scanning of a formula, coupled with the sequential construction of a descriptive formula, are thought of as a simulation of the process by which those abstract maps direct attention.

Also, neurophysiological evidence seems to show that the abstract maps just mentioned are stored in modules located at the motor areas of the parietal cortex, separated from the modules that recognise particular shapes, which are located at ventral areas of the brain (Colby and Merriam, 2005). It seems, as well, that the visual knowledge of the ventral area is organised in accordance with the asymmetric geometry (viewpoint) of the known shapes (Tanaka, 2003, 1997). Hence, the modular set up of GGTS.pl, and the classification of the symbols in accordance with their asymmetric geometry are understood as a partial representation of the functional anatomy of the brain areas dedicated to *high-level vision* (Wandell et al., 2005).

Fourth, linguistic studies have shown that the use of prepositional expressions suggests that human mind abstracts spatial relations with the aid of a topological geometry in which the location of an object is relative to the location of other objects in the scene (Talmy, 2001; Herskovits, 1997). Thus, describing an image as an assemblage of basic features, and the use of prepositions to describe the abstract connections sustainable by any pair of parts conforming a whole, are considered to be a plausible representation of the *mental language* that allows human cognition to perceive coherent wholes made up of discernible parts.

If those correspondences are accepted, then it could be thought that GGTS.pl gives a plausible functional account of the basic mechanisms used by the human mind during the processing of visual representations, including the design of *novel valuable images*. Having said that, GGTS.pl allows to argue that creativity is not an exclusive feature of productive activities, but an essential element of perception. Many of the cognitive abilities used during the production of novel objects are used as well during the recognition of valuable and novel objects. In the case of El Tajín relieves, both, sculptors and spectators, most probably employed a set of compositional conventions internalised as abstract maps. Sculptors probably used these maps to select the position of each shape, and to plan the movements of their limbs. Observers, on the other hand, probably utilised the same maps to plan the movements of their eyes and to check if every feature was located in the correct position. Likewise, sculptors and spectators probably had to explore a space of possibilities before deciding whether a particular combination of features correctly instantiated an accepted convention. Both, also, probably had to compare a well formed object with previous creations in order to know if a totally new object was witnessed.

8 Discussion

As a cognitive functional model, GGTS simulates four aspects of visual knowledge: 1)the application of a topological geometry, 2)the use of a mereological ontology organised in accordance with the asymmetric geometry of the forms conforming visual experience, 3)the flow of the process known as *active vision*, and 4)the design of symbols. The first two aspects are embodied by the symbols of the system and the modular organisation of the productions. The other two are simulated by the derivation process of the grammar: the most abstract schemata predict the layout of a complex scene and activate a chain of modules that predict the shape of the features composing complex wholes. When the aim is the production of a formula, the corroboration of every foreseen consequence guarantees the quality of the object; and when the task is to give sense to a formula, the confirmation of anticipated outcomes explains away many possibilities assuring the plausibility of the final categorization.

As a creative mechanism, GGTS is a well-known tool. Boden (1998) and Wiggins (2005) classified generative grammars as exploratory mechanisms and subordinated them to transformational systems, which not only explore structured conceptual spaces, but distort their boundaries in order to discover new compositional possibilities. Accordingly, GGTS only explores a fixed conceptual space.

Notwithstanding this limitation, a shape grammar can still bring about some surprises. Generative grammars implemented in Prolog generate variations with backtracking as its only aid. However, in the case of GGTS.pl, a slight improvement was obtained by introducing two sets: one for the storage of valuable products (*KV*), the other for the storage of rejected products (*KR*). With this set up, the grammar not only generated valuable objects, but was able to detect the novel ones, even when the formulas where not created by it, or were generated during independent sessions. Other programs have used analog solutions. The problem with GGTS is that it makes this evaluation only on finished objects, where as other programs do it on earlier stages of the production process (Pérez y Pérez and Sharples, 2004; Pérez y Pérez, 2007).

Like most generative grammars, GGTS cannot extrapolate general rules from particular instances. The only way to expand its knowledge base is by manual introduction of information. However, it is worth to consider that the plain addition of features (terminal icon names) in the appropriate categories creates a whole new set of novel and valuable complex images.

Despite its negligibility, the afore mentioned improvements are useful, once applied in the proper situation. In many occasions, archaeologists and art historians advance competing definitions without giving a mechanical procedure for the testing of their theories. In cases like these, the exploratory creativity of shape grammars might be the objective tool that could evaluate the predictive power of competing style definitions. As any other archaeological theory, every time new sculptures were associated to indubitable archaeological contexts, each style definition would have the opportunity to be tested, being better the one accepting the biggest number of legitimate cases and rejecting the biggest number of unacceptable objects. Accordingly, a shape grammar could also be considered a tool for scientific discovery. Every time a legitimate archaeological finding could not be recognized by the best grammar, this finding would have to be considered a real scientific discovery, since the best theory did not predict it and scientists were forced to revise and, perhaps, reject earlier inferences (Aliseda Llera, 2006). Finally, a shape grammar with the sets KV and KR, is a catalogue capable of accepting only new instances of a certain style. This catalogue, given the parsing mechanism of the grammar, would also be capable of interpreting and sorting out the images stored in it, assisting archaeologists not expert in iconography in the elaboration of accurate field reports.

A final archaeological potential is related to the expanding conditions of a grammar. If the set of terminal symbols (*T*) was transgressed, so that some features were incorrectly sorted, then the new grammar would *create* a set of new but illegitimate variations, that, nonetheless, would look very similar to well formed images. In this case, teachers training archaeologists would have a mechanical way to produce expert appraisal drills.

At this point, two shortcomings of GGTS.pl should be reminded. First, it does not process pixel patterns. The user would have a better experience if she/he could avoid the writing of the formulas. Second, the productions of the system could be more flexible, enabling the designer to introduce a smaller number of them. These flaws are expected to be surpassed in subsequent versions of GGTS.

To summarise, GGTS and GGTS.pl have the limitations known for every generative grammar, but this condition did not forbid the generation of things that were not there from the beginning. Also, it should be reminded that slight changes enhanced the performance of a classical formal system. Therefore, GGTS and GGTS.pl show that generative grammars, if put in a convenient context, are still useful instruments for scientist and students.

Acknowledgements

The authors give thanks to the Instituto Nacional de Antropología e Historia (INAH) for its permission (401-3-4042) to reproduce the tagged relieves presented along the paper. We acknowledge that this permit only allows nonprofit uses and does not donate any kind of copyright over the images, for the objects reproduced are property of the Mexican Nation.

Part of Álvarez Cos' contribution was funded by a CONACyT scholarship (August 2004 to June 2006).

References

Aliseda Llera, A. (2006). *Abductive Reasoning: Logical Investigations into Discovery and Explanation*, volume 330 of *Synthese Library*. Springer, Dordrecht/ New York.

Bajcsy, R., Wu, C. K., and Wang, D. Q. (1984). Visual and conceptual hierarchy: A paradigm for studies of automated generation of recognition strategies. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5:319–325.

Bellanger, L., Husi, P., and Tomassone, R. (2006). Statistical aspects of pottery quantification for the dating of some archaeological contexts. *Archaeometry*, 48(1):169183.

Boden, M. (1998). Creativity and artificial intelligence. *Artificial Intelligence*, 3:347–356.

Camiz, S. (2004). On the coding of archaeological finds. *Archaeologia e Calcolatori*, 15:201–218.

Carmena, J. and Nicoleli, M. (2003). Learning to control a brain-machine interface for reaching and grasping by primates. *Public Library of Science. Biology*, 1(2):193–208.

Castillo-Peña, P. (1995). *La Expresión Simbólica de El Tajín.* INAH, México.

Colby, C. and Merriam, E. P. (2005). Active vision in parietal and extrastriate cortex. *Neuroscientist*, 11(5):484–493.

Cowan, N. and Moorey, C. C. (2006). Visual working memory depends on attentional filtering. *Trends in Cognitive Science*, 10(4):139–141.

Dawson, P. C. and Richard, L. (2005). Using computer modelling and virtual reality to explore the ideological dimensions of thule whalebone architecture in arctic canada. *Internet Archaeology*, 18:2–3.

Ebert, D. (2006). Predictive modelling and time: An experiment in temporal archaeological predictive models. *Internet Archaeology*, 20:2–3.

Henderson, J. M. and Ferrerira, F. (2004). *The Interface of Language, Vision and Action. Eye Movements and the Visual World.* Psychology Press, New York.

Herskovits, A. (1997). *Language, Spatial Cognition and Vision.* Stock, Oliviero (ed.) Spatial and Temporal Reasoning, Kluwer. Dordrecht.

Kampen, M. (1972). *Sculptures of El Tajn, Veracruz.* University of Florida Press, USA.

Kirsch, J. L. and Kirsch, R. A. (1988). The anatomy of painting style: Description with computer rules. *Leonardo*, 21(4):437–444.

Matsello, V., Kijko, V., Masuch, H., and Stanke, G. (2002). Recognition of mason marks images. *Proc. of Inter. Conf. Electronic Images and Visual Arts EVA*, pages 69–76.

Mushiake, H., Fuji, N., and Taji, J. (1999). Microstimulation of the lateral wall of the intraparietal sulcus compared with the frontal eye field during oculomotor tasks. *Journal of Neurophysiology*, 81:1443–1448.

Pascual Soto, A. (1990). *Iconografía Arqueológica de El Tajín.* Fondo de Cultura Econmica, México.

Pascual Soto, A. (2005). *El Tajn. En busca de los orígenes de una civilización.* IIE-UNAM-INAH, México.

Pérez y Pérez, R. (2007). Employing emotions to drive plot generatioin in a computer-based storyteller. *Cognitive Systems Research*, 8(2):89–109.

Pérez y Pérez, R. and Sharples, M. (2004). Three computer-based models of story telling: Brutus, minstrel and mexica. *Knowledge-Based Systems*, 17(1):15–29.

Preucel, R. W. (2006). *Archaeological Semiotics.* Blackwell, New York.

Prinke, A. (2005). Digitising historic excavation archives. *Internet Archaeology*, 18.

Proskouriakoff, T. (1953). *The Classic Art of Central Veracruz.* University of Texas Press, Austin.

Renfrew, C., Ezra, B., and Zubrow, W. (1994). *The Ancient Mind. Elements of Cognitive Archaeology.* Cambridge University Press, Cambridge.

Scidmore, S. and Bajcsy, R. (1979). Computer analysis and description of pottery sherd patterns. *International Journal on Policy Analysis and Information Systems*, 3(1).

Talmy, L. (2001). *Toward a Cognitive Semantics*, volume 1. MIT-Press, Cambridge.

Tanaka, K. (1997). Mechanisms of visual object recognition: Monkey and human studies. *Current Opininion in Neurobiology*, 7(4):523–529.

Tanaka, K. (2003). Columns for complex clustering of cells with similar but slightly different stimulus selectiveness. *Cerebral Cortex*, 13(1):90–99.

Wandell, B., Brewer, A. A., and Dogherty, R. F. (2005). Visual field map clusters in human cortex. *Philosophical Transactions of the Royal Society, section B*, 360(1456):963–707.

Wiggins, G. A. (2005). Searching for computational creativity. *IJCAI'05 Workshop on Computational Creativity*, pages 2–3.

Wise, A. and Miller, P. (1997). Why metadata matters in archaeology. *Internet Archaeology*, 2.

Tra-la-Lyrics: An approach to generate text based on rhythm

Hugo R. Gonçalo Oliveira
CISUC
Dep. of Informatics Engineering
University of Coimbra, Portugal
hroliv@dei.uc.pt

F. Amílcar Cardoso
CISUC
Dep. of Informatics Engineering
University of Coimbra, Portugal
amilcar@dei.uc.pt

Francisco C. Pereira
CISUC
Dep. of Informatics Engineering
University of Coimbra, Portugal
camara@dei.uc.pt

Abstract

This paper is about an ongoing project, *Tra-la-Lyrics* which aims to create a computer program capable of generating lyrics for given melodies. As a preliminary phase, the correlations between words and rhythm in a song's lyrics was studied and some heuristics were achieved. Algorithms for syllabic division and syllabic stress identification on a word were implemented. A method for calculating the strength of each beat of a melody was also used. The actual system's architecture is described. To get the contents of the lyrics we've used a relational database where we could find not only the words, but also their grammatical category and some morphological related attributes. Some ongoing work is also referred and some examples of the currently possible outputs of the system are presented. Conclusions and possible further work are finally discussed.

Keywords: computational creativity, rhythm, text generation, music, lyrics

1 Introduction

In the last few years we've been starting to accept the computer as a an artificial artist, highly contributing for turning the creative systems an interesting topic for research. There are many known creative systems with purposes like, for example, composing musical pieces [Cope (1987)], telling stories [Bringsjord and Ferrucci (1999); Gervás et al. (2006)], making up jokes [Binsted et al. (1996)], creating visual art [Machado and Cardoso (2000)], making up poetry [Manurung (2004); Gervás (2000); Díaz-Agudo et al. (2002); Gervás (2001)] and so on. During these later years we have also seen a huge growth of systems where users can customize some digital and multimedia contents they can share with their whole community or simply keep them for their own use. Combining these two with the challenge it represents for research and the fun we hope it will provide, our objective is to design and develop a system capable of generating lyrics in the Portuguese language, from given melodies.

Lyrics have strict metrics that gives us the notion of rhythm. Metrics is obtained by the existence of syllables with different strengths. Also very important is the choice of words, which should not only obey the metrics, but also employ some phonetic patterns like the use of rhyme or alliterations. Although we may argue lyrics have always an emergent semantics, the semantic aspects are not a present concern of our project. We may even state that there shouldn't be a predefined semantics at all letting the users have their own free interpretation. Most of the times it is possible to make some interpretation out of any word sequence, specially if the words follow some gramatical structure.

With this system, we hope to contribute to the general endeavor of building artificial artists. It intends to accomplish one of the less explored tasks in automatic music production: writing the lyrics, given the melody. As far as we know, there were no previous attempts to develop such a system.

The structure of this paper starts by referring some important authors and applications in the area of creative text generation, as they can be included in the same category of the system are creating. After presenting some preliminary work, we will introduce the system itself, *Tra-la-Lyrics*. There will be given special focus to its actual architecture, the way it represents some important contents and also some ongoing work. At the end two sample generated lyrics will be shown.

2 State of the Art

2.1 Poetry generation systems

From the existing systems, poetry generating ones are probably the closest to ours. In this subsection we will refer some works in the area of poetry generation and their similarities and contributions to our system.

In his thesis, Hisar Manurung Manurung (2004) proposes an evolutionary approach for poetry generation and provides a computational model of metrical similarity, the minimum edit distance. In this thesis there is a whole chapter dedicated to poetry and automatic poetry generation where he describes some interesting aspects like a definition for poetry, something about the process of creating poetry and a categorization for the existent poetry generation systems. He defines poetry in a way that it can be falsifiable and though a possible topic for research, experiment and evaluation. A poetic text should follow the properties of meaningfulness (M), grammaticality (G) and poeticness (P). It must convey some conceptual message, obey the given grammar rules and have some poetic features such as rhythm and rhyme. According to the properties they are in agreement with, he divides the existing poetry generation systems into four categories: word salad (respects no property); template and grammar-based (respects G property); form-aware (respects either P or G and P property); and poetry generation (respects G, P and M property). The minimum edit distance, used to evaluate the metrics, basically picks a candidate verse and compares its metrics with some target pattern. The distance is calculated by the number of syllables we would have to change so that the candidate verse fits the metrics.

Pablo Gervás has many works in the area. He was one of the first authors to make serious attempts to poetry generation. He created the WASP system Gervás (2000) whose main objective was to study the effect of the initial data in the resulting poem and to test different generating strategies. ASPERA Gervás (2001) and COLIBRI Díaz-Agudo et al. (2002) are examples of semi-automatic poetry generation system that take advantage of case-base reasoning techniques to generate Spanish poetry. In Gervás (2001) it is referred that there are three major challenges concerning automatic poetry generation, respectively the specification of the formal requirements that define a correct poem, the appropriate management of an extensive vocabulary and the correct combination of words.

2.2 Related applications

We can find many online applications able to generate creative text in the *web*. *The Poetry Creator* [Lewis and Sincoff (2006)] is a system based on verse templates, where the user is asked to give a subject, a synonym for the subject and a title for the poem. The generated text includes both the subject and its synonym in the middle of some poetic verses. There seems to be no present notion of metrics or rhymes. As an example we have generated a poem whose subject was *music* and the given synonym was *songs*. We got verses like *"Alms for the poor!" cried the quickly, ecstatic music With lightning strokes, the songs shot foreward;briefly, keenly*. In *The Essay Generator* [Mullen (2006)] generates an essay about some topic the user gives as an input. The generated essay includes not only words but also figures and references. They have always the same structure with an introduction respectively followed by the *Socials Factors*, *Economic Factors*, *Political Factors* and *Conclusions* sections. Nevertheless the essays are most of the times completely nonsense, because they merely include the topic in previously made sentences. As an example we have generated an essay on the topic 'music and lyrics' and got an essay with sentences like *"Difference among people, race, culture and society is essential on the survival of our world, however music and lyrics smells of success."* or *"The question which we must each ask ourselves is, will we allow music and lyrics to win our vote?"*. *SCIGen* [SCIgen (2005)] is an automatic computer science research paper generator, including graphs, figures and citations. The user only gets to include the name of the authors of the paper. The contents of the paper are then generated using a context-free grammar. *SCIGen*'s main objective is to test whether some conferences have low submission standards. The generated paper titled *Rooter: A Methodology for the Typical Unification of Access Points and Redundancy*[1] was once accepted.

Poesybeat [Initiative (2005)] is not a creative text output system, but an interesting collaborative system maintained by an online community. Users can do one of two things. Either they can send poems they would like to have a melody so they could be sung, or they can send melodies they wanted to have lyrics. The community users try then to find suitable choices. The final result is recorded so that everyone can listen to it. We are basically trying to automatize the *search of lyrics for a melody* part.

3 Preliminary study

This section briefly presents some preliminary work. We needed to study about the correlation about the words and the melody notes in the song lyrics.

3.1 Syllables

Common sense tells us that, in a song, each syllable of a word is associated to a note, so we had to find a way of dividing Portuguese words into syllables. In order to accomplish that, we've implemented our own algorithm. Although it is based in the algorithm presented in [Velloso (2006)], which was made for word wrapping in text processors, we had to add many unhandled situations. Our algorithm is based in some special groups of characters, shown in Figure 1.

The original algorithm divides the words into syllables using the groups VOWELS, CONJ1, CONJ2 and CONJ3. Each group of vowels identifies a different syllable. A complete syllable can also include consonants so, each consonant will be included in the previous or next syllable, depending on which of the groups (CONJ1, CONJ2, CONJ3) it belongs to. For example, for the word *reintegrar* we got the division *rein-te-grar*. We have added a second part for the algorithm, that uses the division made in the first part and then checks whether there is a syllabic division in the middle of a group of vowels. This only happens if the group is neither a dipthong nor part of a ('g', 'q') + 'u' + vowel construction. For example. given the division *rein-te-grar*, we get the final division *re-in-te-grar*. In this word, the dipthong "ei" doesn't exist because the letter 'i' is followed by a nasal consonant.

3.2 Lyrics and rhythm

In order to generate lyrics for the given melodies we already knew there were some important rules, suggested

[1]http://pdos.csail.mit.edu/scigen/rooter.pdf

```
VOWELS = ('a', 'e', 'i', 'o', 'u', 'á', 'à', 'ã', 'â', ...);
SEMI_VOWELS = ('i', 'u');
GA = ('á', 'à', 'é', 'í', 'ó', 'ú');
CIRC = ('â', 'ê', 'î', 'ô', 'û');
TIL = ('ã', 'õ');
H = ('h');
U = ('u');
CONJ1 = ('b', 'c', 'd', 'f', 'g', 'p', 't', 'v');
CONJ2 = ('c', 'l', 'n');
CONJ3 = ('l', 'r', 'z');
CONJ4 = ('q', 'g');
```

Figure 1: Groups of characters used in syllabic division

in some online courses like the one from Berklee *Lyric Writing: Writing Lyrics to Music* [Pattison (2002)] and other online sources like [Simon (2006); Demeter (2001)]. Perhaps the most important rule is to make sure that the stressed syllables are associated with the strong beats in a way that the lyrics have the melody's metric [Hayes and Kaun (1996)]. In order to have some support for this rule and to find out some other correlations between the syllables of the lyrics and the beats of the melody we've implemented a simple information extraction program. The program would give us some information particularly for Portuguese. Its input was a set consisting of the melody and the lyrics of 42 Portuguese popular songs with the most usual types of meter[2] and its output was a set of tables where we had information related to the amount of stressed and unstressed syllables occurring in strong beats and also in weak beats. The information we obtained was in agreement with the rule of Bruce Hayes. Additionally, we collected information about the resolutions of weak syllables in strong beats, syllable prolongation and syllable contraction.

3.3 Stressed syllables and strong beats detection

To identify the stressed syllables in the words, we have implemented an algorithm that used the syllabic division and searched for characters and situations that could stress some syllable. Some examples of these situations are the presence of a graphical accent or the word terminating in a vowel followed by an *r*, *l* or *z*. There is also a list of 18 unstressed monossylables: *o*, *a*, *os*, *as*, *um*, *uns*, *me*, *te*, *se*, *lhe*, *nos*, *lhes*, *que*, *com*, *de*, *por*, *sem*, *sob*, *e*, *mas*, *nem*, *ou* [Sílaba (2006)]. All the words that didn't match any of the rules were considered as being stressed in their penultimate syllable. This happens to be a rule for the Portuguese language.

We have used the dots system, present in *A Generative Theory for Tonal Music* [Lerdhal and Jackendoff (1983)] in order to get the strength of the beats present in the most usual types of meter. Considering level 0 the strongest, we obtained the results shown in Figure 2 in what concerns to the syllables present in the 42 songs. We divided each note/syllable pair into three categories: notes with stressed syllables, notes with unstressed syllables and linked notes. Linked notes are usually associated with a prolongation of the syllable in the previous note and are as strong as the beat in the note they are linked to, so we could consider them as neither stressed nor unstressed syllables.

[2]We have considered the following meter types: 2/4, 3/4, 4/4, 3/8 and 6/8

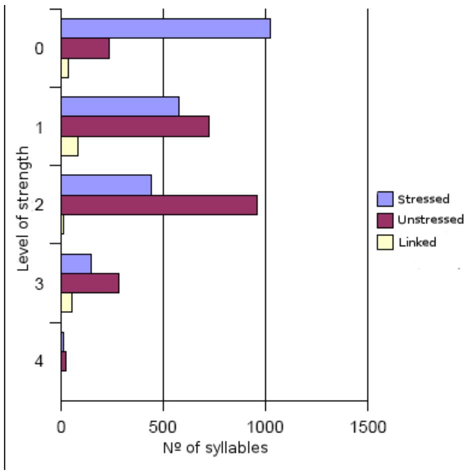


Figure 2: Distribution of syllables per level of strength

3.4 Rhymes and Repetition

As suggested in *How to Write Lyrics* [Demeter (2001)], there are three important points one should pay attention when choosing words for a song: **rhythm**, **rhyme** and **repetition**. We have already referred how we handle the rhythm. In order to provide some rhymes in the right places, we have added the possibility of giving the program a list with the notes where we want them to occur. Each melody can have its own list. Rhymes are get by placing words with the same termination in the given locations. In a later version we intend to automatize or assist the identification of these locations by using a melody segmentation analysis similar to the one in [Grilo (2002)]. Repetition has a very tight relationship with rhythm and it is often present in visual art and music. When it comes to repetition, we have simply added a component that lets us control the probability of reusing previously chosen words, if they fit.

4 Tra-la-lyrics: System description

4.1 System architecture

Our system is implemented in the *Java* programming language and uses *MySQL* as a database engine. In Figure 3 it is possible to see the generic architecture of Tra-la-Lyrics' current version.

4.1.1 Inputs and Output

The basic input of our system is a MIDI file, which is converted into the ABC notation [Gonzato (2003)] using an external tool like *midi2abc*. After a couple of stages, suitable text for the given rhythm will be generated. At the end, we'll have a new ABC file, consisting of the original one with the generated lyrics inserted. Using tools like *abc2ps* and *ps2pdf* we can easily get PDF sheet music from the ABC file.

4.1.2 The modules

We briefly explain each one of the modules of our system:

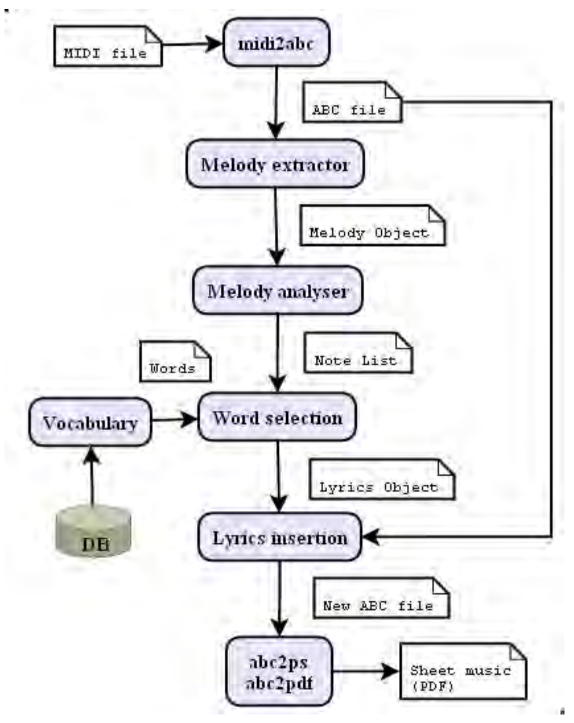
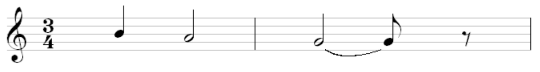
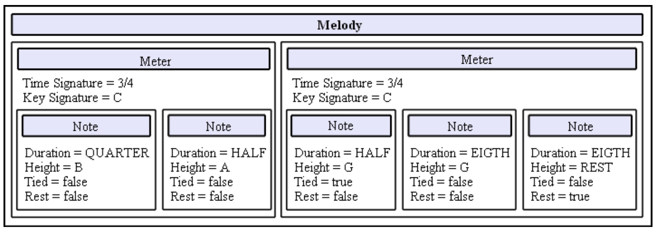


Figure 3: Generic system architecture





Melody

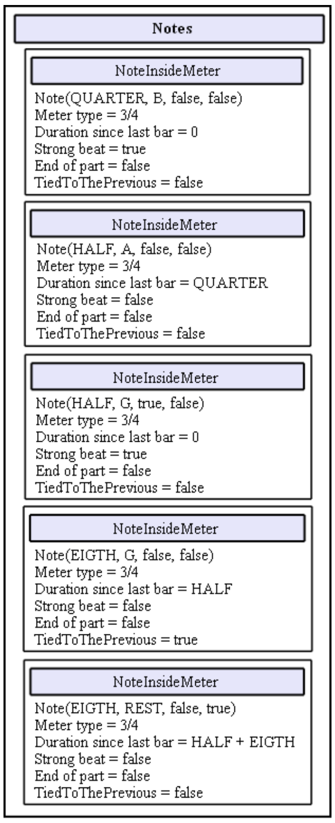
Meter		Meter		
Time Signature = 3/4 Key Signature = C		Time Signature = 3/4 Key Signature = C		
Note	Note	Note	Note	Note

Duration = QUARTER
Height = B
Tied = false
Rest = false
	Duration = HALF
	Height = A
	Tied = false
	Rest = false
		Duration = HALF
		Height = G
		Tied = true
		Rest = false
			Duration = EIGTH
			Height = G
			Tied = false
			Rest = false
				Duration = EIGTH
				Height = REST
				Tied = false
				Rest = true

Figure 4: The *Melody* Object for the above sample notes

- **Melody extractor:** this module gets information from the ABC file, using the *ABC4J API* Gueganton (2005) and builds a *Melody Object* (Figure 4), which is basically a list of *Meters*. The *Meters* themselves are lists of *Notes*.

- **Melody analyzer:** for each *Note* of the melody, a new object *NoteInsideMeter* is created. It contains not only the *Note* object, but also its position inside the meter, its strength and whether it is a note corresponding to the end of a part and thus suitable to be the start of a rhyme[3]. We get the strength of the notes using the dots system present in *A Generative Theory for Tonal Music* [Lerdhal and Jackendoff (1983)]. The output of the *Melody analyzer* is an object consisting of a list of *NoteInsideMeter* objects and some methods to get the distance (number of notes) between a given note and the next where a **special pattern** occurs. We call this object *Notes* and its rep-



Notes

NoteInsideMeter
Note(QUARTER, B, false, false)
Meter type = 3/4
Duration since last bar = 0
Strong beat = true
End of part = false
TiedToThePrevious = false

NoteInsideMeter
Note(HALF, A, false, false)
Meter type = 3/4
Duration since last bar = QUARTER
Strong beat = false
End of part = false
TiedToThePrevious = false

NoteInsideMeter
Note(HALF, G, true, false)
Meter type = 3/4
Duration since last bar = 0
Strong beat = true
End of part = false
TiedToThePrevious = false

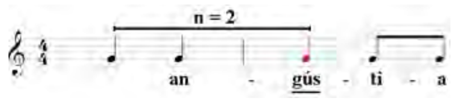
NoteInsideMeter
Note(EIGTH, G, false, false)
Meter type = 3/4
Duration since last bar = HALF
Strong beat = false
End of part = false
TiedToThePrevious = true

NoteInsideMeter
Note(EIGTH, REST, false, true)
Meter type = 3/4
Duration since last bar = HALF + EIGTH
Strong beat = false
End of part = false
TiedToThePrevious = false

Figure 5: The *NoteInsideMeter* Object

resentation for the melody in Figure 4 is shown in Figure 5. The **special patterns** will be listed shortly. Each one of them is an event sequence relevant for fitting the metrics of the text in the metrics of the melody.

- **Word selection:** This module is responsible to get words that fit in the melody rhythm. The *Notes* object is iterated and after calculating the **distance** from the current note to the next **special pattern** a word should be obtained from the *Vocabulary* module. If we get a new word, the next note's index will be needed. It is got by adding the number of syllables of the new word plus the number of rests and the number of linked notes found to the current index.
  In the following we are listing all the considered **special patterns**. For each one of them, different kinds of words will be needed, depending on the calculated **distance** *n*. The words are supplied by the *Vocabulary* module.
  - **Strong beat**: A note occurs in a strong beat. We have marked as strong, the beats whose calculated strength is the highest of the meter or higher than some predefined value.

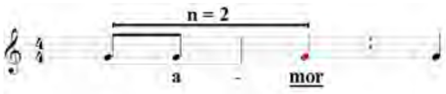
    The word should have at most *n*+2 syllables[4]

[3]In the current version this info is provided by the user.

[4]Portuguese words are stressed in one of their last 3 syllables

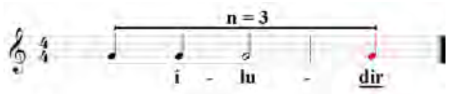
and its stress should be found in the *n*th syllable counting from the beginning of the word. For the example in the picture, we would need a word with 4 syllables with its second syllable stressed, like *an-**gús**-ti-a*.

– **Strong beat followed by a rest**: A note occurs in a strong beat and we can find a rest after it.



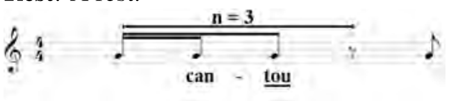
The word should have *n* syllables, and its stress should be found in the *n*th syllable, counting from the beginning of the word. For the example in the picture we would need a word with 2 syllables with its last syllable stressed, like *a-**mor***.

– **Strong beat followed by the end of the melody**: The note occurs in a strong beat and it's the last note of the melody.



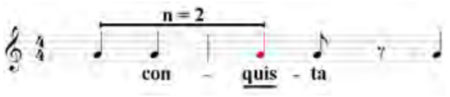
Similar to the previous pattern. For the example in the picture, we would need a word with 3 syllables, with its stress appearing in the last one, like *i-lu-**dir***.

– **Rest**: A rest.



The word should have at most *n-1* syllables, and should not have stress at all. We have only considered 18 unstressed words, all of them monosyllables, making it difficult to have long sequences of unstressed words. However it does not appear to be critical to have a few stressed words not in strong beats.

– **Strong beat followed by some note, followed by a rest**: The note is a strong beat, followed by some other note, after which we can find a rest

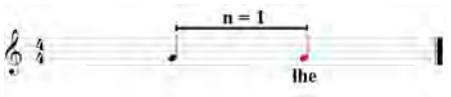


The word should have at most *n+1* syllables and its stress should be found in its *n*th syllable. For the example in the picture, we would need a word with at most 3 syllables, with its 2nd syllable stressed, like *con-**quis**-ta*.

– **Last strong beat of the current part**: The note is in the last strong beat of the current part of the music.

The word should have at most *n+d* syllables, where *d* is the distance between the last strong beat of the part and the last note of the part. The *n*th syllable must be the stressed one.

– **End of the melody**: The note is the last of the melody.



The word should be one of the 18 unstressed monosyllables, because this pattern means we have the last note of the melody and it's not in a stressed beat.

After having the lyrics for the whole sequence of notes, a *Lyrics Object* is built, consisting of a sequence of *Words*, which on their own are sequences of *Syllables*.

- **Vocabulary:** this module returns words, with a maximum number of syllables and a fixed distance to the stressed syllable. The words are obtained from a database, consisting of tagged words for the Portuguese language and some of their syllabic attributes like the syllabic division or the stress position. The method used to get the words can be reimplemented so that we can easily test different strategies.

- **Lyrics insertion:** at this stage the information in the original ABC file is merged with the generated lyrics, giving rise to a new ABC file.

4.1.3 The Database

As a source of words, we have used at first a database consisting of the words in the treebank *Floresta Sintá(c)tica*, which has a great amount of grammatically tagged sentences taken from the Portuguese newspaper *Público*. This treebank can be found in *Linguateca* [Linguateca (2000)]. Due to some mistagged words and an unfriendly database structure (with some information we wouldn't need), we have created a new database with words taken from *Floresta Sint(c)tica*, tagged with a tool called *Jspell* [Jspell (1995); Dias de Almeida and Pinto (1995)]. Everytime we want, we can use a simple program we have implemented to complement this database with words taken from other texts (poetry for instance). The new database keeps information about the grammatical categories of the words, their root, their gender, their number and also specific information about verbs or personal pronouns, like tense or person.

A table containing some syllabic related attributes for the words, like syllabic division, stressed syllable position or word termination was included in both databases. In order to get those attributes, the algorithms referred in Sections 3.1 and 3.3 were used.

4.2 Ongoing work

4.2.1 Strategies to get the words

As said before, the *Vocabulary* module has a method used to get words, based on both the position of their stressed syllable and the maximum number of syllables. This module keeps the strategy of getting words, making the *Words Selection* module simpler and the management and testing of different strategies easier. At the moment we have implemented three strategies:

- **Strategy 1, random words + rhymes:** where the *Vocabulary* simply returns random words and tries to have rhymes in the end of some given parts. The rhymes are simply based on the words' termination.
- **Strategy 2, words following sentence templates + rhymes**: where the *Vocabulary* module has a set of simple Portuguese sentence templates that are basically a sequence of grammar tags. The returned words are forced to follow some randomly chosen templates so that there is some syntactical coherence. There is also a submodule responsible for keeping the gender and the number of the last article or pronoun, forcing the following noun, adjective or verb to have the same morphology. With some probability two templates can be linked using a conjunction.
- **Strategy 3, grammar + rhymes** (currently in development): where the *Vocabulary* module uses a grammar whose derivations build Portuguese sentence templates. Each symbol in these templates is an instance of some grammatical category and has a set of attributes[5] depending on the grammatical category it belongs to. This strategy differs from the above one, specially in the following:
  - Backtracking is used when there are no suitable words to correctly finish a sentence;
  - If a list with the musical parts division is provided, it tries to make each sentence correspond to a musical part;
  - More grammatical categories are used and their attributes can easily be defined;
  - Only open class words[6] are reused.
  - Each production has a different probability of being chosen so that most common sentences are more frequently built that less common ones;
  - It is able to generate a larger amount of different templates;

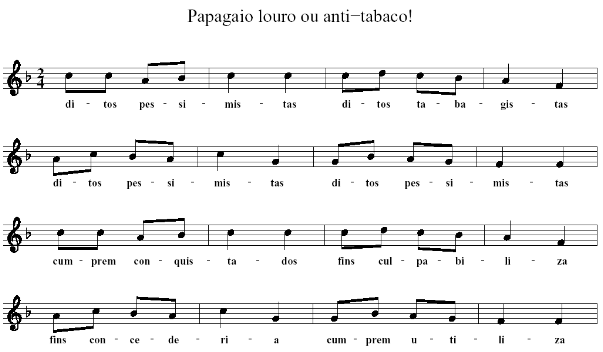
Both strategies 2 and 3 try to get words that rhyme in the end of some parts but, beyond the stress position and the number of syllables, theses strategies have to deal with restrictions like the category of the word and its morphological attributes, making it sometimes impossible to find words that rhyme. In order to minimize this problem, the program does the following: 1) Each time it needs to get a word for the end of a part to start a rhyme, it tries to choose between the matching words with the most common terminations. This maximizes the probability of finding a words that rhymes for the following verse(s); 2) Everytime we end a verse with a different termination, we store that termination. If there no words that follow all the restrictions and rhyme with the previous verse, we try to find words that rhyme with other terminations we had stored.

4.2.2 Finding suitable locations for rhymes

At the moment, it is possible to include in the ABC file a list with with some numbers corresponding to the notes

[5]gender, number, person...

[6]nouns, adjectives and verbs.


Papagaio louro ou anti–tabaco!
di - tos pes - si - mis - tas di - tos ta - ba - gis - tas
di - tos pes - si - mis - tas di - tos pes - si - mis - tas
cum - prem con - quis - ta - dos fins cul - pa - bi - li - za
fins con - ce - de - ri - a cum - prem u - ti - li - za

```
Ditos pessimistas
ditos tabagistas
ditos pessimistas
ditos pessimistas
cumprem conquistados
fins culpabiliza
fins concederia
cumprem utiliza
```

Figure 6: Sample generated lyrics using strategy 1

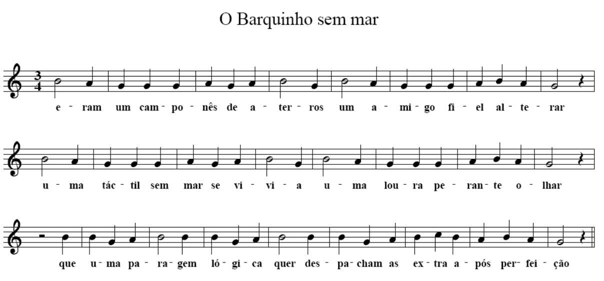
where we'd like to have a rhyme. We are planning to include an evolutionary algorithm similar to the one used in Grilo (2002) so that the suitable places to have rhymes are automatically proposed.

5 Demo runs

In this section we will ilustrate our system's behaviour with the analysis of three generated lyrics. Each one of them uses one of the previously referred strategies. The probability of reusing previously chosen words was set to 20% in the first and second and to 60% in the third. This probability is however only applied when the Vocabulary module is queried for a word and there are suitable words that have already been used in the same lyrics.

- The music shown in Figure 6 is a known Portuguese popular song called *Papagaio Louro*. The lyrics were generated by our program using **Strategy 1**. The lyrics are perfectly matching the rhythm and many words like *ditos* and *pessimistas* have been reused creating a somehow funny effect. We can find rhymes in the desired places (*tabagistas* and *pessimistas*, *culpabiliza* and *utiliza*). However, the words sequence doesn't obey grammatical rules.

- The music shown in Figure 7 is a known Portuguese popular song called *O Barquinho*. The lyrics were generated by our program using **Strategy 2**. Once again the lyrics match the rhythm, even though this strategy has more restrictions to get the words making the probability of not matching the rhythm a little higher. This example has also a rhyme in the right place (*alterar* and *olhar*). The most different aspect they have relative to the lyrics generated with **Strategy 1** is that the words follow some simple sentence templates. The lyrics we present follow templates like:


O Barquinho sem mar
e - ram um cam - po - nês de a - ter - ros um a - mi - go fi - el al - te - rar
u - ma tác - til sem mar se vi - vi - a u - ma lou - ra pe - ran - te o - lhar
que u - ma pa - ra - gem ló - gi - ca quer des - pa - cham as ex - tra a - pós per - fei - ção

```
Eram um camponês de aterros
um amigo fiel alterar
uma táctil sem mar se vivia
uma loira perante olhar
que uma paragem
lógica quer
despachem as extra
após perfeição
```

Figure 7: Sample generated lyrics using strategy 2

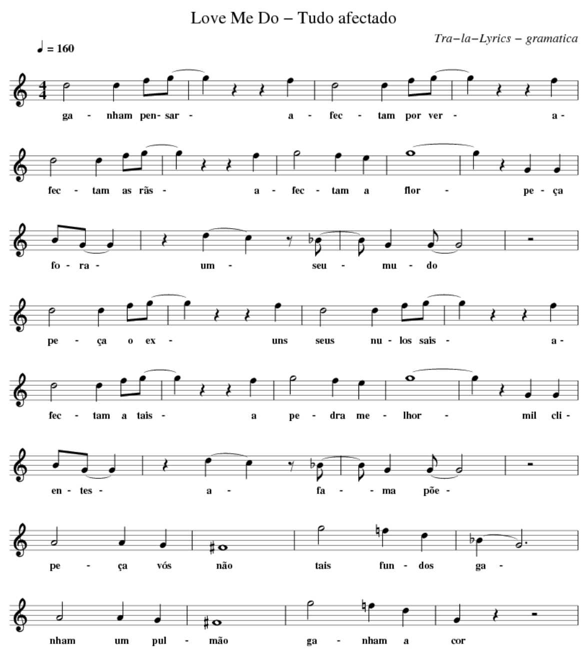
- verb article adjective preposition noun:
  eram um camponês de aterros
- article noun adjective:
  um amigo fiel

The probability of getting a conjunction between two templates was set to 80%, but the only present conjunction is the word *que*. Although we could get some meaning from the lyrics generated with both strategies, there is no explicit semantics. **Strategy 2** simply generates (almost) grammatically correct sentences.

- The music in Figure 8 is a well known song by *The Beatles*, *Love Me Do*. The shown lyrics were generated using **Strategy 3**. They match the rhythm and follow a large amount of templates, like:

  - verb preposition verb:
    afectam por ver
  - verb article noun:
    afectam as rãs
  - article possessive-pronoun adjective noun:
    uns seus nulos sais
  - article noun adjective:
    a pedra melhor
  - article noun verb:
    o fama põe

  This music is divided into many small parts and that's why the lyrics have only short sentences. We can find some rhymes like *sais* and *tais* or *não* and *pulmão*. We can also find the repetition of words like *peça*, *afectam* and *ganham*. The probability of reusing previously chosen words was set to a higher value (60%) because: 1) it is more difficult to find used words that match not only syllabic restrictions, but also category and morphology restrictions; 2) this strategy only tries to reuse words of open classes, as it is said Section 4.2.1. Like the lyrics generated with **Strategy 2**, these have no explicit semantics, however it


Love Me Do – Tudo afectado
Tra–la–Lyrics – gramatica
♩ = 160
ga - nham pen - sar - a - fec - tam por ver - a -
fec - tam as rãs - a - fec - tam a flor - pe - ça
fo - ra - um - seu - mu - do
pe - ça o ex - uns seus nu - los sais - a -
fec - tam a tais - a pe - dra me - lhor - mil cli -
en - tes - a - fa - ma põe -
pe - ça vós não tais fun - dos ga -
nham um pul - mão ga - nham a cor

```
Ganham pensar afectam por ver
afectam as rãs afectam a flor
peça fora um seu mudo
peça o ex uns seus nulos sais
afectam a tais a pedra melhor
mil clientes a fama põe
peça vós não tais fundos
ganham um pulmão ganham a cor
```

Figure 8: Sample generated lyrics using strategy 3

is easier to build up some meaning out of them, because they follow grammatical rules.
This strategy has still a wide range for improvement.

6 Conclusions and further work

Despite the results already achieved we hope to get better ones after refining some details and add new functionalities. We are planning to let the users change some settings, like the strategy used to generate the lyrics, the probability of reusing words or the places where they'd like to have rhymes. It would also be interesting to give them the possibility of having a set of preferred words or even texts, which would have more probability of being included in their generated lyrics.

In the future, more strategies for fitting the words in the rhythm or to get words can be tested. One different approach to fit the words could be similar to the one used by Hisar Manurung in Manurung (2004). We could analyze the melody and create its metric pattern that would be our target. We could then have a population of possible lyrics. To evaluate each lyrics in the population we could use the minimum edit distance, which would give us the amount of syllables not matching the target pattern.

Another strategy for obtaining the words will soon be tested, using the portuguese version of a (still) unpublished surface text realizer. Although the current version of the system only generates portuguese lyrics, it won't be difficult to change it to other languages. The syllabic division and syllabic stress detection algorithms would have to be reimplemented and a different source of words would be needed for each language.

As we all know, creative systems are not easy to evaluate. We are thinking in three different ways of evaluating our system. For testing if the words correctly fit in the rhythm, we can pick up a set of sample generated lyrics, use the information extraction system implemented during the preliminary work, and analyze the obtained information. We will be able to compare the syllabic distribution in our lyrics with the one in the "real" song lyrics. To evaluate the quality of the general output the better way would probably be to have a considerable amount of people answering questionaires, where they would be asked to compare existing lyrics with some generated, both for the same song. Another possible way of evaluating the lyrics' quality could be the implementation of some system that would analyze a set of generated lyrics for the same melody, and using some heuristics, evaluate the presence of poetic features and vocabulary diversity.

Another interesting thing we could do in a near future would be using some sung voice synthesis software, like Singing Computer [Zamazal (2001)], to sing our generated lyrics. Singing Computer uses Lilypond music notation files as input and there is one known ABC to Lilypond converter (abc2ly), making our task easier.

References

Binsted, K., Pain, H., and Ritchie, G. (1996). *Machine humour: An implemented model of puns*. PhD thesis, University of Edinburgh.

Bringsjord, S. and Ferrucci, D. A. (1999). Artificial intelligence and literary creativity: Inside the mind of brutus, a storytelling machine. Lawrence Erlbaum Associates, Hillsdale, NJ.

Cope, D. (1987). An expert system for computer-assisted music composition. *Computer Music Journal 11,4 (Winter)*.

Demeter (2001). How to write lyrics. http://everything2.com/index.pl?node=How%20to%20write%20lyrics.

Dias de Almeida, J. J. a. and Pinto, U. (1995). Jspell - um módulo para a análise léxica genérica de linguagem natural. http://natura.di.uminho.pt/~jj/pln/jspell.ps.gz.

Díaz-Agudo, B., Gervás, P., and Gonzlez-Calero, P. A. (2002). Poetry generation in colibri. *ACM - Digital Library*.

Gervás, P. (2000). Wasp: Evaluation of different strategies for the automatic generation of spanish verse. *Proceedings of the AISB00 Symposium on Creative and Cultural Aspects and Applications of AI and Cognitive Science*.

Gervás, P. (2001). An expert system for the composition of formal spanish poetry. *Journal of Knowledge-Based Systems*, 14:181–188.

Gervás, P., Lönneker-Rodman, B., Meister, J. C., and Peinado, F. (2006). Narrative models: Narratology meets artificial intelligence. In Basili, R. and Lenci, A., editors, *International Conference on Language Resources and Evaluation. Satellite Workshop: Toward Computational Models of Literary Analysis*, Genova, Italy.

Gonzato, G. (2003). The abc plus project. http://abcplus.sourceforge.net.

Grilo, C. F. A. (2002). *Aplicação de Algoritmos Evolucionários à Extracção de Padrões Musicais*. PhD thesis, University of Coimbra.

Gueganton, L. (2005). abc4j. http://gueganton.chez-alice.fr/abc.

Hayes, B. and Kaun, A. (1996). The role of phonological phrasing in sung and chanted verse. *The Linguistic Review*.

Initiative, P. A. (2005). poesybeat. http://poesybeat.org/.

Jspell (1995). Jspell. http://natura.di.uminho.pt/natura/natura?topic=jspell.

Lerdhal, F. and Jackendoff, R. (1983). *A Generative Theory of Tonal Music*. The Massachussets Institute of Tecnhology, 2nd edition - 1996 edition.

Lewis, J. and Sincoff, E. (2006). Poetry creator 2. http://www-cs-students.stanford.edu/~esincoff/poetry/jpoetry.html.

Linguateca (2000). Linguateca. http://www.linguateca.pt.

Machado, P. and Cardoso, A. (2000). Nevar - the assessment of an evolutionary art tool. http://citeseer.ist.psu.edu/machado00nevar.html.

Manurung, H. (2004). *An evolutionary algorithm approach to poetry generation*. PhD thesis, University of Edinburgh.

Mullen, D. (2006). Essay generator. http://radioworldwide.gospelcom.net/essaygenerator/.

Pattison, P. (2002). Lyric writing: Writing lyrics to music. http://www.berkleeshares.com/download/870472/berklee_musical_stress.pdf.

SCIgen (2005). Scigen - an automatic cs paper generator. http://pdos.csail.mit.edu/scigen/.

Sílaba (2006). A sílaba. http://www.geocities.com/shiurtalmid/catan/portuguese/v6.html.

Simon, T. (2006). Criterios para relacionar letra e musica. http://www.musicaeadoracao.com.br/tecnicos/musicalizacao/letra_musica.htm.

Velloso, A. T. (2006). Separando slabas com c#. http://www.microsoft.com/brasil/msdn/Tecnologias/visualc/SeparandoSilabas.mspx?mfr=true.

Zamazal, M. (2001). Singing computer. http://freebsoft.org/singing-computer.

Session 3
Musical Creativity

A Hybrid System for Automatic Generation of Style-Specific Accompaniment

Ching-Hua Chuan[*] **and Elaine Chew**[†]
University of Southern California Viterbi School of Engineering
[*]Department of Computer Science and
[†]Epstein Department of Industrial and Systems Engineering
Integrated Media Systems Center, Los Angeles, CA
{chinghuc,echew}@usc.edu

Abstract

Creating distinctive harmonizations in an identifiable style may be one of the most difficult tasks for amateur song writers, a novel and acceptable melody being relatively easier to produce; and this difficulty may result in the abandonment of otherwise worthwhile projects. To model and assist in this creative process, we propose a hybrid system for generating style-specific accompaniment, which is capable of creating new harmonizations for melodies, with proper harmonic resolutions, in a style that is learned from only a few examples. In the proposed system, a chord tone determination module first learns, then determines, which notes in a given melody are likely chord tones. According to these chord tones, triads are assigned first to the bars with unambiguous solutions, and these triads serve as checkpoints. The system then constructs possible chord progressions using neo-Riemannian transforms between checkpoints, and represents the alternate paths in a tree structure. A Markov chain with learned probabilities for these neo-Riemanian transforms then generates the final chord progression. We select four songs by the British rock band, Radiohead, to evaluate the system. Three songs are used for training, and an accompaniment is generated for the held out melody. We present the results of two case studies. We find that the system generates chords closely related to the original, and the resulting chord transitions reinforce the phrase structure of the melody.

Keywords: Automatic Style-Specific Accompaniment, Chord Tone Determination, Neo-Riemannian Transforms, Markov Chains.

1 Motivation

In this paper, we describe an automatic style-specific accompaniment system that makes song writing accessible to both experts and novices. This work is inspired by the fact that many people without formal musical training can sing karaoke, some even quite well, but have difficulty crafting sophisticated chord arrangements in specific styles for the melodies which they sing with such ease. Without proper accompaniment, those creative melodies would have only a limited existence, and would probably soon be discarded. Thus, our solution to this problem is to create a system that would automatically generate accompaniment to a melodic composition in a specific style, given exemplar pieces.

Such a system must satisfy the following requirements. First, the system should be able to identify the features important to the style specified by the user, based on only a few examples. For music novices, it may be difficult for them to use musical terms to describe a style, but it is intuitive for them to ask for harmonization similar to some particular songs. Second, the system must be capable of creating new chords not present in the example pieces, but these chords should still be consistent with the specified style. Third, chord transitions and harmonic resolutions must follow the style of the examples. Last but not least, the accompaniment needs to support the phrase structure of the melody, for example, through the insertion of proper cadences at phrase endings.

In this paper, we propose a hybrid system for generating style-specific accompaniment. The system combines music theoretic knowledge and statistical learning, which has the advantage of being able to simultaneously maintain stylistic elements, such as chord tones and chord transitions, learned from the examples, and create new chords with the theoretically and structurally correct harmonic resolutions. Statistical learning allows the system to construct style-related rules for automatic accompaniment, however, the approach becomes problematic when there are only limited numbers of training examples. The specification of style is often best done with no more than a few examples, as large numbers can dilute the defining features. However, rules learned from only a few examples cannot be widely generalized. In particular, the generating of new chords sequences with appropriate transitions are especially difficult for a purely statistical learning system. Furthermore, without music knowledge, such as the use of cadences at phrase endings and of neo-Riemannian transforms that ensure parsimonious voice leading, a sequential and statistical approach has difficulty generating appropriate chord progressions with proper structural emphases.

The system comprises of several modules employing different computational approaches. The system first determines the chord tones in the melody. The chord tone determination module applies machine learning techniques to choose the chord tones from the input melody based on the example pieces. The system uses seventeen attributes to represent the melody, including phrase structure information. Then, chords are prescribed at checkpoints in the melody where it has been determined that the harmony in that bar is unambiguous. Using these checkpoints as anchors, we use neo-Riemannian transformations to build chord progressions between between consecutive checkpoints, according to the chord transitions learned, and making sure to provide correct and smooth harmonic resolutions. Finally, we use Markov chains to generate the final chord progression. Section 2 presents the details of each of these system modules.

To demonstrate and evaluate the system, we train the system on three songs of the British rock band, Radiohead, and generate chord progressions for the fourth. The original accompaniment of the fourth piece serves as ground truth. Section 3 presents the results of the experiment and system evaluation.

1.1 Related Work

Automatic accompaniment as a harmonization problem has been studied for more than a decade. Natural Language Processing techniques, such as n-gram statistical learning, have been applied to the learning of musical grammars for harmonizing music in the style of the seventeenth century (Ponsford et al., 1998). Evolutionary techniques, such as Genetic Algorithms, have been proposed and implemented in the generating of four-part chorales (Phon-Amnuaisuk et al., 1999). Phon-Amnuaisuk and Wiggins (1999) compared the results between genetic algorithms and a rule-based system for solving a four-part harmonization problem, and found that the rule-based system performed better than the one employing genetic algorithms. Other techniques, such as Hidden Markov Models, have also been utilized in the harmonization of chorales (Allan and Williams, 2005).

Chord progressions, transitions, and resolutions in the harmonization of pop-rock music have also been studied by musicologists and music theorists. Instead of using traditional roman numerals, Kochavi (2002) and Capuzzo (2004) used the neo-Riemannian framework for analyzing the chord transitions in pop-rock music as transformations on the *tonnetz*, thus demonstrating a viable representation and robust grammar tool that accounts for tonal ambiguities caused by modal mixture in pop-rock music.

Most music computational methods adopt sequential processing in time, i.e., processing the note/bar that occurs earlier before processing the next note/bar. When sequential processing is not required, we have the advantage of selecting the process order in a way that best benefits the task at hand. In the proposed system, instead of computing sequentially, we assign chords first at the checkpoints, then determine the chord progressions in the bars between. Similar computational ideas can be found in Chew and Wu (2005), where the separating of pitches into different voices is done through the connecting of maximal voice contigs, where the solution is certain.

2 System Description

This section provides the details of the proposed hybrid system for automatic accompaniment. The system and data flow diagram is shown in Figure 1. The system consists of three major parts performing the following tasks: chord tone determination, triad construction and checkpoints setup, and chord progression generation. Chord tone determination, described in Section 2.1, chooses the chord tones from the melody. Based on the chord tones reported by the previous module, triads are first used to harmonizing the parts of the melody that contain notes identified strongly as triadic chord tones. The details for triad assignment are shown in Section 2.2.

The third part of the system, described in Section 2.3, is responsible for generating the chord progression for the entire melody. Based on the triads assigned in the previous module. All possible progressions between any two of these triadic checkpoints are generated by applying neo-Riemannian transforms. We use a tree structure to represent the possible paths, and a pruning algorithm to remove paths containing disallowed transitions. The final chord progression is created by considering the conditional probabilities of all possible paths as represented in a Markov chain.

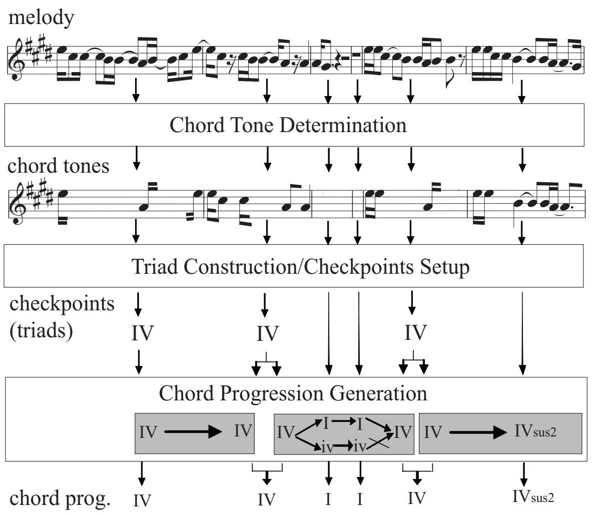


Figure 1: The data flow and system modules for automatic accompaniment

2.1 Chord Tone Determination

The chord tone determination module classifies notes in each bar into chord tones and non-chord tones. We separate this module from the next one, chord determination, for a few reasons: chord tone classification becomes simpler when one does not consider the relations between adjacent chords; this module also learns the melodic harmonization style of each bar; and, new chords (chords not in the training examples) can still be constructed in a consistent style. For example, a sus4 chord can still be produced by adding the 4th note to the basic triad if the 4th note is reported as a chord tone, even if the sus4 chord does not appear in the training data.

We apply machine learning techniques to determine the chord tones. This module consists of a Support Vector Machine (SVM), which is responsible for selecting chord tones from the melody in each bar in order to learn the types of melodic harmonization decisions made typically by the particular band based on the examples provided. In the learning stage, each note in a bar is represented by the seventeen attributes described in Section 2.1.1; the ground truth is provided by the rhythm guitar chords in the original lead sheet. In the testing stage, the module classifies a note as a chord tone or non-chord tone based on the seventeen attributes of the melodic representation.

The output of the chord tone determination module is a list of notes that should be considered chord tones in each bar of the test piece. Given the output list, a few possible chords can be chosen as harmonization candidates.

2.1.1 Melody Representation

We use seventeen attributes to describe each melody note in a bar. The meanings of these attributes are shown in Table 1. We represent each pitch as a numeric pitch class, numbered from zero to eleven, and normalized so that the tonic is zero. The duration represents the length of the pitch class in beats. The next four attributes describe the scale relationships among the pitch classes in the bar, and include the presence of the upper and lower neighbors, the third, and the fifth. We only consider the intervals of thirds and fifths because neo-Reimannian operations are based on triads, even though they have been extended to include seventh and ninth chords transitions (see Kochavi (2002), Capuzzo (2004).)

Table 1: Attributes for melodic representation

Attribute	**Meaning**
Pitch class (pc)	numeric pc, tonic normalized to zero
Duration	note duration (in beats)
Upper neighbor	pc one scale step above present?
Lower neighbor	pc one scale step below present?
Third	pc three scale steps above present?
Fifth	pc perfect fifth above present?
Metric strength 1	note on metric strength level 1?
Metric strength 2	note on metric strength level 2?
Metric strength 3	note on metric strength level 3?
Metric strength 4	note on metric strength level 4?
Metric beat 1	note on beat 1?
Metric beat 2	note on beat 2?
Metric beat 3	note on beat 3?
Metric beat 4	note on beat 4?
Phrase Position	start, middle, end or bridge
Number of pc's	total number of pc's in bar
Odd/Even bar	note in odd or even bar in phrase?

The next eight attributes relate to metric information. The metric strength attribute shows the metric strength (frequency of onsets on that position in the bar, reminiscent of the inner metric calculations described in Volk (2002) and Chew et al. (2005)) of the pulse on which the note resides. The metric beat attribute records the metric position in the bar according to the time signature. The phrase position in the training data is manually annotated as: start, middle, end, or bridge (an interlude segment between two phrases.) The last two attributes provide information on the number of pitch classes in the bar, and the whether the bar count is odd or even within the phrase.

2.2 Triad construction and checkpoints setup

With the list of chord tones in each bar, we assign triads to harmonize each bar. Triads are the basic building blocks for more elaborate chords. In triad assignment, we appoint chord tones with attributes that strongly support their being members of a triad, such as chord tones having their Third present in the same bar, or chord tones having their Third and Fifth in the same bar. If a chord tone can be harmonized by a major as well as a minor triad, chord selection is determined based on the conditional probabilities calculated during the learning stage.

By using the selected chord tone checkpoints, we first determine the chords for the bars with strong evidences for harmony choice independently of the bars with less evidence for harmonicity. A cadence is a typical example of such a checkpoint. These checkpoints act as the stable points for starting the harmonization process. A wrong chord at a checkpoint dramatically reduces the quality of the song's accompaniment. For instance, incorrect harmonies at a cadential checkpoint can easily result in auditory discomfort.

The setting up of checkpoints divides the harmonization task into smaller sections of chord series generation. Instead of finding a chord progression path for the entire melody, we generate a suitable path of chord progression between each pair of the adjacent checkpoints. This setup not only makes the computation efficient, but also enables us to break the sequential order for processing music. For example, cadential checkpoints help to ensure proper chord resolutions at phrase endings.

2.3 Chord progression generation

This section describes the assignment of chords at the chord tone checkpoints, and between checkpoints.

2.3.1 Chord candidate selection

When the choice for triad assignment is clear, we set the accompaniment chord for that bar to this triad. If the chord tones cannot be harmonized by any one triad, an extended seventh chord will be considered. If the chord tones cannot be fully covered by any triads nor seventh chords, then a compound chord, consisting of pitches from a cluster of chords, will be constructed in order to cover as many of the chord tones in the bar as possible.

A compound chord is constructed based on distance (number of neo-Riemannian transforms) in the chord space shown in Figure 2. For example, the chord pair (I, vi) is a cluster of two chords with a neo-Riemannian distance of one. The cluster having the shortest total inter-chord distance is chosen as the compound chord, and the function of this chord is determined later based on the context; the determination of context will be described in Section 2.3.2. If there are multiple clusters with the same shortest distance that cover all the chord tones in a bar, then all possibilities are kept as potential candidates.

The new chord construction mentioned above extends the system's vocabulary of chords so that it is capable of harmonizing the melody using new chords not in the learning examples. The system can also generate chords that are neither triads nor seventh chords, but are chords that can be frequently found in popular music, if the chord tones contain sufficient information. For instance, the Fsus4 chord, consisting of the pitches {F, B♭, C}, can be covered by the cluster (F, B♭), i.e., (IV, ♭VII) in the key of C major, and is functionally regarded as a subdominant in chord transforms. The extra note, B♭, will not sound incongruous, as long as it is resolved correctly, according to neo-Riemannian transformations.

2.3.2 *Neo-Riemannian transforms*

Neo-Riemannian transforms have been used by music theorists to analyze harmonic patterns and voice-leading in pop-rock music in the recent decades (Kochavi, 2002; Capuzzo, 2004). There are four fundamental operations in neo-Riemannian transforms for describing the chord progressions: I (Identity, same chord), L (Leading-tone exchange), P (Parallel), and R (Relative), as shown in Figure 2. Although the biggest benefit provided by neo-Riemannian transformations is modal music analysis, we still represent chords in terms of roman numerals in this paper for the following reasons: melodies sung by people are, for the most part, still tonal music; and, the roman numerals normalize all pieces by key, thus reducing the number of pieces required in the learning stage.

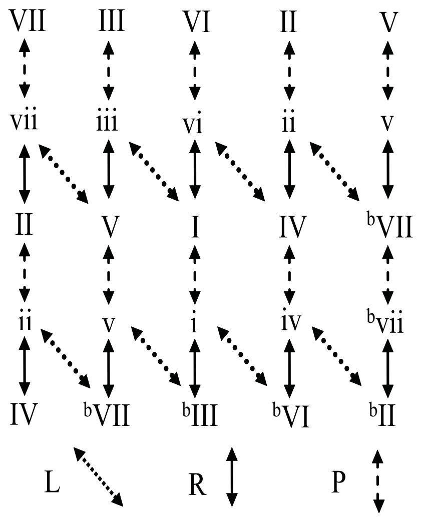


Figure 2: Neo-Riemannian transforms in the chord space.

2.3.3 *Tree construction and pruning*

We use a tree structure to represent the possible chord progressions between two checkpoints. In the tree, each node represents a chord that contains all the chord tones generated for that bar. A child node represents a chord that results from a learned neo-Riemannian transform to the next bar. The height of the tree equals the number of bars between and including the two checkpoints.

To construct a valid tree for chord progressions between two checkpoints, three constraints must be satisfied. The first two are local constraints, while the third one is a global constraint: (1) the chord selected in each bar should contain all the reported chord tones; (2) a mapping between two adjacent chords must be a valid (learned) neo-Riemannian transform; and, (3) the root node chord must be the first checkpoint, and a leaf node chord at the bottom of the tree must be the second checkpoint.

If a branch cannot continue grow to the second checkpoint, then the branch would not produce a plausible progression. We apply a pruning algorithm for removing those stunted branches in order to make the chord progression calculations more efficient. The pruning algorithm works as follows: if a node cannot establish a valid link to any of the nodes in the next level, it is considered a dead end, and it will report this information to its parent. If a node, n, receives a "dead end" message from all of its children, then n becomes a dead end too. The pruning process continues backtrack until it reaches either a node containing a live child or the root.

An example of a tree structure for chord progressions is shown in Figure 2.3.3. The roman numeral at a node shows the chord candidate for the bar at that level, based on the reported chord tones for that bar. The circled nodes are the checkpoints, which are harmonized by the I chord in this example. Links between nodes (shown with arrows) are created based on valid neo-Riemannian transforms; the particular neo-Riemannian operations invoked are shown in italics. The dashed arrows represent pruning actions when a node cannot continue to grow to the next level. In this example, a total of three valid chord progressions are found: {I, I, V, V, I}, {I, I, iii, iii, I}, and {I, I, iii, vi, I}.

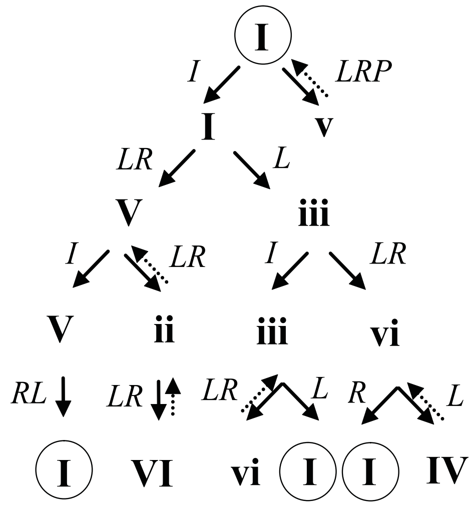


Figure 3: Example tree structure for chord progression.

It may be possible that no branches can be found that connect the two checkpoints, due perhaps to errors in chord tone determination or to a sparse learning set. In this situation, the generation of chord progression will be split into two sub-problems. At each split, we allow an arbitrary *non*-Reimannian transition. A possible heuristic for selecting a split point considers the number of types of neo-Riemannian transforms learned in bar transitions. A bar transition having more possible chord transforms is more likely to allow more flexible (possibly new) harmonizations (transitions), and is thus a good candidate for a split point. The worst case splitting occurs when the chords selected in each bar cover the chord tones, but no neo-Reimannian transform exist to transition between the bars.

2.3.4 *Markov Chains*

After constructing the tree, we can readily extract all successful paths between the checkpoints. Each of these paths can be considered a Markov chain, and the likelihood of that path can be calculated from the conditional probabilities in the Markov chain.

Assume we have a path with n chords, $\{C_1, \ldots, C_n\}$, where each chord is indexed by its bar number. The probability that this chord progression occurs can be expressed as:

$$\begin{aligned} & P(C_1, \ldots, C_n) \\ & \quad = \quad P(C_1)P(C_2|C_1)\ldots(C_n|C_{n-1}) \\ & \quad = \quad P(C_1)P(NRO_{1,2})\ldots(NRO_{n-1,n}), \end{aligned} \tag{1}$$

where $NRO_{i-1,i}$ is the neo-Riemannian operation between chord C_{i-1} and C_i. Equation 1 accounts for the probability of the chord progression, but it ignores the phrase information for each bar. In order to generate a chord progression that better reflects the phrase structure of the melody, we modify Equation 1 to include the phrase position information of each bar:

$$\begin{aligned} & P(C_1, \ldots, C_n|B_1, \ldots, B_n) \\ & \quad = \quad P(C_1|B_1)P(C_2|C_1, B_1, B_2) \\ & \quad\quad\quad \ldots P(C_n|C_{n-1}, B_{n-1}, B_n) \\ & \quad = \quad P(C_1|B_1)P(NRO_{1,2}|B_1, B_2) \\ & \quad\quad\quad \ldots P(NRO_{n-1,n}|B_{n-1}, B_n) \end{aligned} \tag{2}$$

where B_i is the phrase position for bar i, which falls into one of four possible categories: start (S), middle (M), end (E), and bridge (B), as described in Table 1. For example, $P(LR|S, M)$, the probability that neo-Riemannian operations LR occurs from a starting bar to a bar in the middle of the phrase, can be calculated from the examples as follows:

$$P(LR|S, M) = P(LR, S \to M)/P(S \to M) \tag{3}$$

The first term, $P(C_1|B_1)$, in Equation 2 is redundant if C_1 is a checkpoint at the first bar. When C_1 does not occur at the first bar, we may have multiple choices for the chord there, and the term $P(C_1|B_1)$ does affect the resulting progression. If the conditional probability of all chord candidates are zero, due to limited learning pieces, we substitute the term $P(C_1|B_1)$ with $P(C_1)$ instead.

3 Evaluations and Results

In this paper we test our system on music by the British rock band, Radiohead. The experiment design and results are detailed in the following sections.

3.1 Experiment design

We consider four songs by Radiohead: *Creep*, *High and Dry*, *Fake Plastic Trees*, and *Airbag* for our accompaniment generating experiment. We consider these four songs similar, not in terms of particular musical features such as melody, chord progressions, and rhythmic patterns, but according to general properties, for example, all four songs were published in their first three albums, they are each relatively slower than other songs in the respective albums, and each song shows a clear acoustic rhythm guitar accompaniment. However, we did consider one musical fact: all four songs are in major keys; we considered this fact important because strategies of chord progressions would be very different in major vs. minor keys.

We obtained the lead sheets for the songs from the website *http://www.gprotab.net*. For each score, we extracted the two main tracks, melody and rhythm guitar, and removed all other instruments. We manually verified the chords, selecting the main chord for each bar, and discarding ornamental chords and other musical elaborations for the purpose of training and evaluation. Phrase information for each bar and key information are also added to the annotation. The repeats in each song are truncated to simplify the process.

The number of appearances of chords along with their phrase position $\in$ {S, M, E, B} of the four songs are shown in Figure 4a through 4d, where chords are represented as roman numerals, i.e. their function within the key. The phrase position M is furthered specified as being an odd or even bar, {Mo, Me}. Notice that the choices of chords and their distributions are very different from one song to another.

To test our system, we held out one song as the test melody, and used the remaining three training examples. For training, the melodies (represented as described in Section 2.1.1) as well as the chords are given to the chord tone determination module for the learning of the harmonization strategy. The conditional probabilities of chord appearances and neo-Riemannian transform are also calculated from the training examples at this stage. For testing, only the melody is given to the chord tone determination module. Based on the reported chord tones, a chord progression is generated according to the process described in Section 2.3.

3.2 Case Study 1: Creep

First, we choose *Creep* as the test song, and trained the system using the other three. Using the original accompaniment for *Creep* as ground truth, the overall correct rate is 81.48% in this 54-note sample. The statistics on the chord tone determination, such as true positive rate (TP), false positive rate (FP), precision (Prec.), recall (Rec.), and F-measure (F), are shown in Figure 5. Notice that the false positive rate is much lower than the true positive rate. Thus, by generating accompaniment according to the reported chord tones, the harmonization process should still result in the original chords.

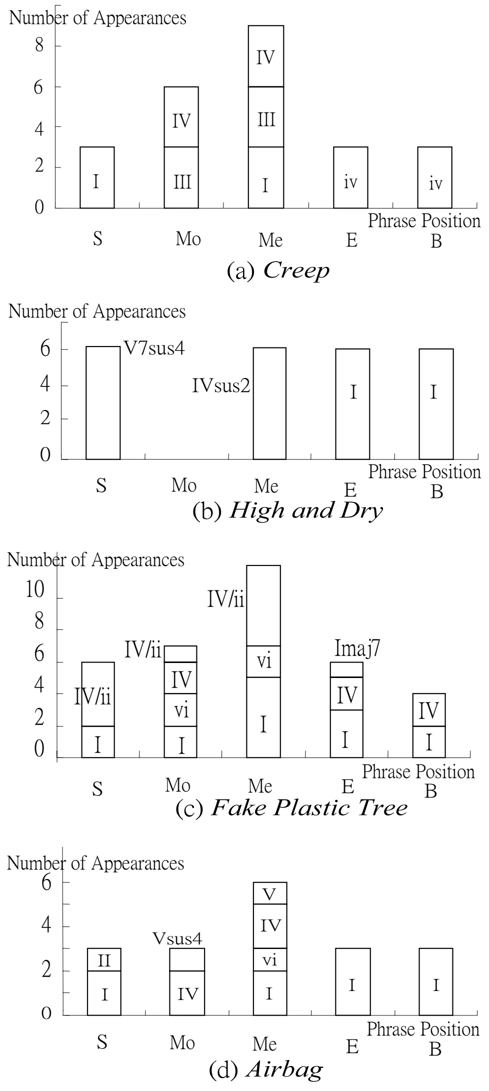


Figure 4: Chord distributions in the original songs

Figure 6 shows the generated chords as well as the original harmonizations for *Creep*. In the 26 bars, 9 of the generated chords are identical to the original. Most of these chords occur at the start of phrases beginning with a I chord, or in the middle of a phrase with IV chord. In the remaining bars, 5 generated chords are the parallel major/minor of the original ones, and 3 generated chords are related by a fifth to the original ones.

With regard to chord tone determination, there are only two false positives, pitch A in bars 1 and 13. In bar 1, the reported chord tones are G, A, which can be harmonized by the cluster of chords IV and the ii of IV in the chord space shown in Figure 2. This generated compound chord, IV+2, still has the function of IV, and is successfully resolved by the LR operation to the I chord in the next bar. In bar 13, the reported chord tones are B, A, harmonized by the iii+2 chord, which happen to be a seventh chord in this case.

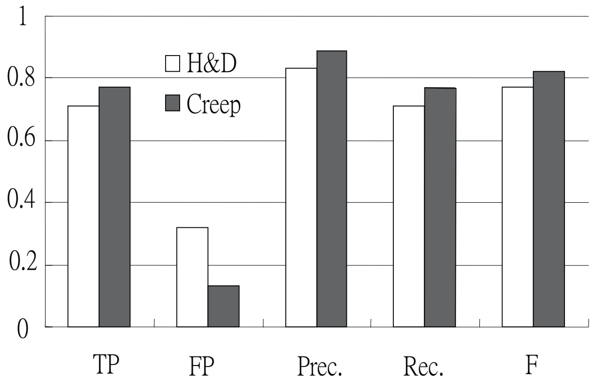


Figure 5: Chord tone determination statistics – true positive (TP), false positive (FP), precision (Prec.), recall (Rec.), and F-measure (F) – for *Creep* and *High and Dry*.

The original harmonization contains only four chords, repeated periodically in every phrase. The chord B (III) and Cm (iv) are missing in the generated chords. Instead, their parallel major/minor are chosen. We can explain the result by examining the chord transitions in the training songs. In Figures 4b, 4c, and 4d, no chords are related by parallel major/minor in each song. Therefore, no parallel (P) neo-Riemannian operations are learned at the training stage. Although the chord iii and IV are more commonly used than III and iv, and no non-scale pitches such as D♯ and E♭ appear in the melody, the generated chords here may lack some of piquant harmonies of the original. For the bars labeled as cadences (bars 8, 16, 24, and 26), the chords (G, C, G, G) are generated instead of (Cm, Cm, Cm, G) as in the original.

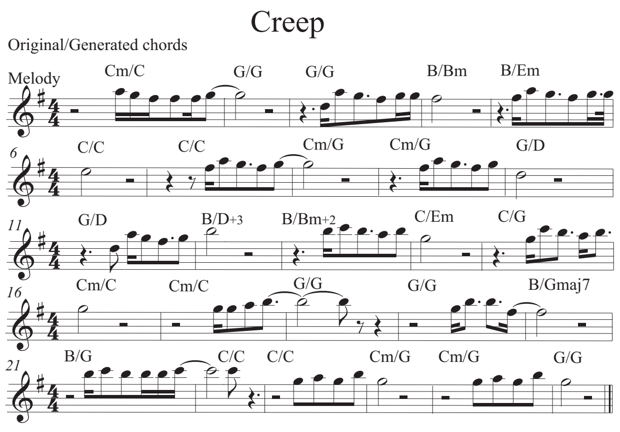


Figure 6: Original/Generated chords for *Creep*.

3.3 Case Study 2: High and Dry

We choose *High and Dry* as our second test piece. The overall correct rate, when comparing to the original lead sheet, is 70.49% on this 61-note sample. The statistics in

Figure 5 show that chord tone determination fared better in *Creep* than in *High and Dry*, especially for the false positive rates.

Three types of false positives are found in the chord tone determination. The first type occurs in bars 2 and 10, where the pitch C♯ is wrongly reported as a chord tone. This results in the choice of the chord A instead of Asus2. The second type happens in bar 22, where the pitch G♯ is reported falsely as a chord tone. However, the resulting compound chord E+4 includes all the pitches in Asus2. The third type occurs in bars 16 and 18. The wrongly reported chord tone F♯ results in a humdrum but straightforward chord progression from bars 16 through 18.

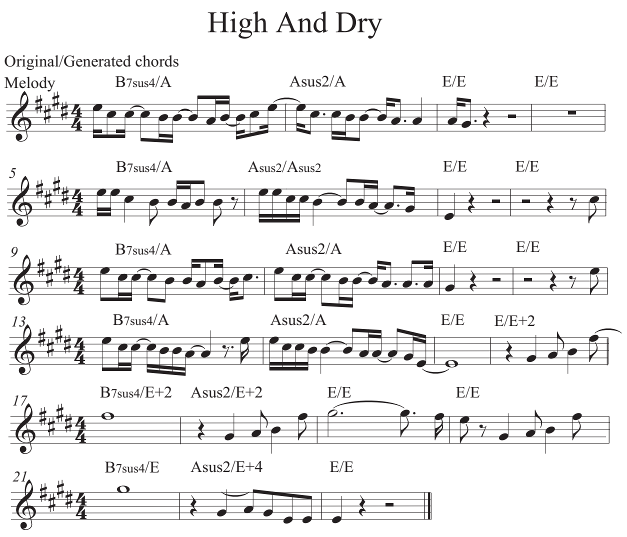


Figure 7: Original/Generated chords for *High and Dry*.

Figure 7 shows the generated chords as well as the original harmonization of the song *High and Dry*. In the 23 bars, 11 of the generated chords are identical to the original. An additional other 6 of generated chords either show the same basic function (bars 2, 10, 14, and 16) or cover the pitches in the original chords (bars 18 and 22). The original harmonization shows regular four-bar chord pattern: B7sus4 → Asus2 → E → E. A similar structure can be observed in the generated chord progressions: A → A → E → E. The chord E (I) is generated for all the bars labeled as cadences (bars 3, 7, 11, 15, and 23), as in the original song.

4 Conclusions and Discussion

In this paper, we proposed and demonstrated a hybrid approach to the building of an automatic style-specific accompaniment system. The system aims to make song writing accessible to novices and experts alike. Implementation details of song writing such as chord assignment and arrangement often requires years of musical training and practice, and may prevent less experienced song writers from focusing on the expression of their musical ideas. Our objective is to design a system that can not only provide proper chord progressions to melodies created by novices, but also present new harmonization ideas to experts. To this end, we have proposed a system that models the harmonization process in a sequence of logical steps, and generates chords with proper resolutions. The system is designed not only to allow users to concentrate on higher level decisions and to focus on creative ideas, it also serves as a systematic model for the process of generating accompaniment.

In the two test examples, we demonstrated the system's ability to create new chords, while maintaining the proper chord transition as in the provided examples, in accordance to the phrase structure of the melody. We find that the system generates chords closely related to the original, and the resulting chord transitions meet our expectations based on the melodic phrase structure. We also observed a few challenges in the chord generating process. For example, the melody of *Creep* does not contain strong cues for harmonization, and the system harmonized the same melody in different bars using different chords. Sometimes, one may choose to include chord tones that do not appear in the melody to improve voice-leading, or for ease of fingering on the instrument (e.g. guitar tablature); this are aspects which are currently not captured in the system. Neo-Riemannian operations are based on triads, and are not flexible for generating chords such as IVsus2 with exact pitches. Finally, symmetric phrase structures, reflected for example by regularly repeated chord patterns, are difficult to generate when using only local bar-by-bar analysis. Future systems could incorporate such higher level structures.

Acknowledgements

This research has been funded by the Integrated Media Systems Center, a National Science Foundation (NSF) Engineering Research Center, Cooperative Agreement No. EEC-9529152, a University of Southern California Women in Science and Engineering (WiSE) Digital Dissertation Fellowship, and by the NSF grant No.0347988. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors, and do not necessarily reflect the views of NSF or WiSE.

References

Allan, M. and Williams, C. K. I. (2005). Harmonising chorales by probabilistic inference. In Saul, L. K., Weiss, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems: Proceedings of the Neural Information Processing Systems Conference 2004, Vancouver, B.C.*, volume 17, pages 25–32. MIT Press, Cambridge, MA.

Capuzzo, G. (2004). Neo-riemannian theory and the analysis of pop-rock music. *Music Theory Spectrum*, 26(2):177–199.

Chew, E., Volk, A., and Lee, C.-Y. (2005). Dance music classification using inner metric analysis: a computational approach and case study using 101 latin american dances and national anthems. In Golden, B., Raghavan, S., and Wasil, E., editors, *The Next Wave in Computing, Optimization and Decision Technologies: Proceedings of the 9th INFORMS Computer Society Con-*

ference, volume 29 of *Operations Research/Computer Science Interfaces*, pages 355–370. Springer, Annapolis, MD, US.

Chew, E. and Wu, X. (2005). Separating voices in polyphonic music: A contig mapping approach. In Wiil, U. K., editor, *Computer Music Modeling and Retrieval: Second International Symposium, CMMR 2004, Esbjerg, Denmark, May 26-29, 2004, Revised Papers*, volume 3310 of *Lecture Notes in Computer Science*, pages 1–20. Springer-Verlag, Berlin, Germany.

Kochavi, J. (2002). *Contextually Defined Musical Transformations*. PhD thesis, State University of New York at Buffalo, Buffalo, New York.

Phon-Amnuaisuk, S., Tuwson, A., and Wiggins, G. (1999). Evolving music harmonisation. In Dobnikar, A., Steele, N. C., Pearson, D. W., and Albrecht, R. F., editors, *Artificial Neural Nets and Genetic Algorithms: Proceedings of Fourth International Conference in Portoroz, Slovenia*. Springer, Vienna, New York.

Phon-Amnuaisuk, S. and Wiggins, G. (1999). The four-part harmonisation problem: A comparison between genetic algorithms and a rule-based system. In *Proceedings of Society for the Study of Artificial Intelligence and Simulation of Behaviour Convention*, Edinburgh, Scotland.

Ponsford, D., Wiggins, G., and Mellish, C. (1998). Statistical learning of harmonic movement. *Journal of New Music Research*, 28(2):150–177.

Volk, A. (2002). A model of metric coherence. In *Proceedings of the 2nd Conference on Understanding and Creating Music*, Caserta, Italy.

On the Meaning of Life (in Artificial Life Approaches to Music)

Oliver Bown
Centre for Cognition, Computation and Culture,
Goldsmiths College, University of London,
New Cross, SE14 6NW, UK
o.bown@gold.ac.uk

Geraint A. Wiggins
Centre for Cognition, Computation and Culture,
Goldsmiths College, University of London,
New Cross, SE14 6NW, UK
g.wiggins@gold.ac.uk

Abstract

Artificial life (alife) is of interest to computer musicians due to its generative potential and the potential for producing lifelike behaviours for musical interaction. In this paper we consider how future developments in alife music could have equal bearing on the major themes in alife as on the music it produced. We focus on a discussion of the socio-cultural dimensions of making music with technology and argue that modern popular music making practice outside of individualist academic research projects is an important context for the development of alife music systems. This discussion introduces a number of themes about how computational creativity and human creativity may interact as the field progresses.

1 Introduction

Artificial life (alife) and artificial intelligence (AI) exist as independent subjects: put crudely, life does not require intelligence (the intelligence explored by *good old fashioned AI*) (Brooks, 1990), and intelligence (of that same kind) does not require life. Alife as a whole is unambiguously dedicated to the theoretical study of life, and the experimental study of lifelike systems *in silico*. Any notion of alife music (by which we mean composition and performance using alife systems, rather than the scientific study of music as a system of interaction using an alife methodology) lacks this purity of focus; it is a peculiar hybrid. And yet it is also an emerging field, alongside the use of alife in other arts, which sees great potential in the application of broad computational questions of life within artistic practice, including with respect to the mimicking of human creativity. The purpose of this paper is to untangle the divergent goals of alife and regular music practice and to attempt to focus on a potential common interest in terms of a *strong* notion of alife music[1].

It is easy to accept a notion of music *inspired by* alife systems, or music that is generated by alife systems, and it is clear that the patterns generated by such systems are likely to have some degree of musical intrigue, both in their local temporal structure and in their ability to generate variation (Miranda, 2001; Berry and Dahlstedt, 2003). But interesting musical structures can also, according to other music practitioners, be made by monitoring atmospheric conditions, cosmic rays, or by recording the sound of wind-induced vibrations on London's Millennium Bridge. Endlessly generative (thus arguably *creative*) systems can also be produced by less exotic approaches than alife, including the simple combinatoric approaches developed by artists like Brian Eno, for whom a specific kind of musical style, ambient music, was required in order to facilitate *acceptable* generative pieces (Eno, 1996) (also see Jem Finer's *Longplayer*, http://www.longplayer.org). Alife-inspired or alife-generated music has a place in the context of these existing approaches to music, where interesting natural dynamics and powerful generative methods are appropriated to musical ends. In this paper we will be interested in discussing a coming together of alife and music that is stronger than this, in an attempt to approach long term issues in the development of creative systems.

Alife approaches to music can also fall closely in step with more general AI approaches to music in which the designer's goal is to build a system that achieves some more or less precisely specified musical capacity. Alife is concerned with evolutionary and adaptive systems, and adaptivity is closely associated with the essential nature of life because it defines a system that is able to respond to its environment in a beneficial way, and is therefore able to *survive*. We can use artificial evolution or learning techniques to generate systems that function as musical agents, and this is very interesting because in doing so we have defined an artificial environment, and within that environment the criteria for its inhabitants' survival. But this environment is only partly artificial. The 'artificial' of pure alife is an artificial built exclusively to interrogate the real, but in music it interacts and overlaps with the real; the environment of real musicians making real music. This is

[1]By this we mean a strong notion of alife music rather than the notion of strong alife applied to music, although as this discussion unfolds some may have the sense that these are the same.

the starting point for the discussion in this paper. What is *the meaning of life* in these instances?

2 Alife Music in Practice

The first author's interest in alife music stems from an interest in realtime interactive musical agents, in particular agents that are sufficiently complex that they cannot be controlled directly but must be interacted with, where this interaction provokes a sense of engagement that is musically pleasing, that has some of the characteristics of interaction with another live musician. Many commentators have discussed these issues in the emerging domain of live algorithms (*e.g.*, Blackwell and Young, 2004; Collins, 2006).

Strong alife music systems are music systems that music producers and consumers will genuinely feel are autonomous and lifelike, and the above two qualities are, we assume, necessary attributes of such systems. They are also necessarily subjective. A critical question is whether these systems need to emulate human behaviour, or whether there are other modes of behaviour that can evoke a sense of autonomy in musical contexts. Can we have strong alife music whilst bypassing many of the challenges of understanding human intelligence? It is challenging to imagine musical agents that are not human and also do not explicitly mimic human behaviour but that are convincingly autonomous and lifelike: no such thing exists today.

The first author's personal attempts at alife music involve the use of Continuous-Time Recurrent Neural Networks (CTRNNs) (Beer, 1996; Slocum et al., 2000) in live musical performance. The CTRNNs act as simple decision engines in a complex performance patch built in Max/MSP (www.cycling74.com), which pre-processes audio input for feeding into the inputs of the CTRNN and maps the CTRNN's output to generate audio in various ways. So far all of the CTRNNs used are generated by artificial evolution using simple hand-written fitness functions which express the artists own impressions of what would make interesting musical behaviour. This artificial evolutionary environment is very different from the real environment in which the artist actually exists as a musician. Whether explicitly demoing this system or just using it in the context of performance, during performance the system is inevitably squeezed to fit into a musical goal.

Thus it is common to override or tightly constrain the behaviour of a system during a performance. Whilst the simulated environment represents a first approximation of what a musician expects to be good musical behaviour, there is hardly any overlap between the CTRNN's simulated evolutionary environment and the real environment in which the musician acts. The latter is also very many orders of magnitude more complex than the former. It would seem natural to look back over the design of the entire system and ask how one could adjust this design to make the system more successful. This is the point at which a strong alife approach to music must differ fundamentally from other AI approaches.

Consider the iterative process that starts when any music system is first tested in a real performance context. It would be unusual to have pre-specified quantitative elements to measure the system. Instead it is normal to remain open to the possibility that the system had unexpected positive qualities, even if these are extremely modest qualities. In general we can't predict how people will make use of the system, or what they will make of it, especially if the system is intended to be complex and full of surprises.

Two very simple observations have emerged from working with CTRNNs in a performance context. Firstly, a modest unexpected quality: the activity of the network is valuable in live solo laptop performance even if only to produce loosely synchronised activity, meaning activity which has no precise timing or decision-making demands. This is like having an extra set of hands to control some parameters in a laptop performance. This is a relatively unambitious use of the network, not dissimilar to drawing data from environmental conditions or any of the other examples discussed in the introduction. Our point for the time being is that it is a slight *re-appropriation* in terms of how one conceives of the network and how it is used. Secondly, expectations by third party musical performers about what the network can or should do can become problematic. By disguising the network's behaviour as some relatively direct consequence of a human's activity (the hidden activity of the laptop performer), these expectations are no longer relevant, and this has proven useful. This is not to say that the CTRNN has descended to this level of use, in some cases it has been possible to bring it to the fore and to step away from controlling it, whilst in other cases it is more appropriate to subsume its behaviour under other musical goals and activities. Rather, it proves that the CTRNN serves some purpose as a *mere tool*, with just an inkling of something more worthy of the term alife. This demonstrates what we would call the *cybernetic flexibility* of music as a domain of human activity. In the following section we aim to provide some background to the proposal that this flexibility can and should be captured and used to greater effect.

3 Life

In nature we often view organisms as having adapted to their environments through evolution by natural selection. This is widely understood to be a simplification, albeit one that is highly convenient and often sufficiently accurate. In reality all organisms alter the environment for all other organisms, all evolving, all at once in one large open dynamical system. Lovelock's Daisyworld model provides a definitive proof of concept for this point of view (Lovelock, 1979). The model consists of a planet heated by a sun, and two types of daisies with different heat absorption properties and temperature preferences. Running the model shows the relative populations of daisies stabilising in an arrangement in which the local temperature is optimal for each of the daisy types. Daisyworld's virtual daisies could easily have given the impression to naive observers of having adapted to their environment, but in fact they have altered it through a simple thermo-regulatory process.

The implications of considering the effects of this true *coevolutionary* process are now beginning to be explored and will continue to drive theoretical biology into the future. For example, Owings and Morton's (1998) new approach to animal vocal communication depicts a natural history full of organisms (assessors) whose naturally evolved senses have become a focus of exploitation by others (managers); owls, for example, that trick badgers into thinking that they are snakes by mimicking snake sounds (the owls don't need to know anything about snakes, or why they make these sounds). This supersedes the older view that implicitly accepted the evolutionary centrality of an animal's senses. Rather, the senses are reconsidered as a context for evolution elsewhere.

Our planet's biodiversity, the celebrated evidence for nature's own creativity, derives from a divergent evolutionary process in which different species react to each other through evolutionary change, rather than from a process, as in most artificial evolution for engineering's sake, which generally aims for convergence on a best solution; nature holds no such requirement on any of its constituent organisms. It is this coevolving process that in some way drives the increase in complexity evidenced in nature's history, where we see not only ingenious solutions to problems, but the creation of new niches and survival challenges themselves, and gentle shifts of focus from one evolutionary process to the next.

How can we begin to bring these kinds of issues into a consideration of alife in the context of music? If an important fact about nature is that it places no strict demands on what life looks like and how it behaves, then how can we reconcile our desire to embed alife in a context in which our own aesthetic requirements are positively stifling?

There appear to be three potential views on this problem. Firstly, the above concerns are excessive and place too high a demand on what we call alife. Engineering-oriented artificial evolution can produce surprising creative solutions to problems and systems that are complex enough that their operation is rendered opaque. Alife is meaningful even in massively constrained, human-centric situations. Secondly, going the opposite way, the strong ambitions of alife music are genuinely flawed; the aliveness and, by implication, the autonomy of a system is hampered by the constraints of an evolutionary context allowing only one course of action: to produce pleasing music or musical behaviour. Natural evolution would not work under such constraints, and so the notion of alife in the context of music is a weak one at best.

The third view, which we prefer, breaks the stalemate of this opposition. It proposes that human musical practice is capable of providing a suitably rich environment for divergent, open-ended evolution to take place. In aspiring to the environmental freedom of nature we must find ways to allow as much variety and flexibility as possible into our demands for artificial musical systems, neither limited to the taste of one human participant, nor static and unresponsive to the actions of agents. In aspiring to the complexity of nature, we must facilitate countless continued repeated interactions between agents and their environments.

These new requirements can be summed up in the proposal that the application of alife in music has suffered from an individualistic approach. This individualism is manifest in two ways: the individualism of the user, such as in the case of interactive genetic algorithms where a single user is expected to steer an evolving system towards the fulfilment of their musical requirements; and the individualism of the system's purpose, where we assume a musical purpose for our alife system in order to design it.

It may seem that eradicating the second of these two individualisms would ground ones efforts immediately. Is there any sense in producing an alife music system and *then* deciding how to use it? In the following section we suggest that this *is* possible if we look more closely at questions in common music practice and the socio-technological conditions in which music exists.

4 Music Systems, Practice, Sociality and Technology

Strong alife music cannot be about training a system to achieve a pre-specified goal in a pre-specified style. In this case, the less a system's behaviour has been determined by a single individual's expectations, or with respect to a single musical function, the freer it is to take on the properties of an alife music system, rather than an AI or machine learning system. This assertion is strong but not defeatist. However it does jar in various ways with certain instances of the relationship between music, technology, individuals and groups.

To take an example of early work in intelligent music systems, in discussing the role of computers in music performance, Robert Rowe states that "[the elimination of human performers] is undesirable, beyond the purely social considerations, because human players understand what music is and how it works and can communicate that understanding to an audience, whereas computer performers as yet do not." (Rowe, 1993). This is an agreeable statement, except that there is no way of corroborating the simple assertion that "human players understand what music is and how it works". How can one probe this statement further? How do we *know* that it is true, or even what it means? Rowe's "as yet" proposes that computer performers will arrive at a level comparable to human musical understanding. However, the performers that they aim to imitate and perhaps to replace are entangled in a web of relations and social concepts and structures that are suitably versatile as to bring into focus this problem of their understanding, and how it fits with problems of music and understanding in general.

In our opinion the point of view captured here (which we do not mean to associate explicitly with Rowe, but see as a general view about how computational creativity is likely to unfold) does not truly acknowledge the diversity and strength of difference in approaches to music, especially in new social and technological contexts. As Rowe was writing this, the technology behind the tape music approach that he was contemplating was escaping the academic computer music world and initiating the biggest revolution in popular music since rock and roll. Contemporary Western dance music (encompassing genres such as techno, house, garage, drum and bass, hardcore, dubstep and many others) presents problems for any performance-centric view of music. Much dance music is ‘hand programmed’ by its producer and at no point during its production or consumption does a performance take place. And yet this is extremely expressive music[2]. If there are clear examples of the widespread social acceptance of non-performed music, even if these examples are based on new technology not possessing a long-standing tradition, then the issue of performance must be understood as a non-essential musical element. For some this may be a relatively minor and unproblematic statement, but it remains an outside point of view and one that is rarely stressed.

But going deeper into the difference between live performed music and studio-based composition, the problem of autonomy in computer music systems becomes centralised through a notion of *editorship*. Imagine a human-edited recording of a piece of computer generated music; let us assume that you enjoy the piece very much, but you have no idea what work was done by the human during the editing phase. If your aim is to judge the musicality of the artificial system, you will find this opacity of presentation naturally quite unsatisfactory: judging a system implies judging it *in action*. For this reason, live performance provides a context in which the evaluation of computer music systems seems to make greater sense. In reality the same problems of editorship still apply. We have described this in the first author’s performance work above; the editorship of any software activity generally takes priority. But even in systems that are not tampered with during or after playback, human premeditation and planning are still largely responsible for the ultimate aesthetic and content of the music. Furthermore, through focusing on the musical performance a false boundary arises around the beginning and end of a single performance event. We judge individual human musicians over their careers, and only some musicians in some contexts are valued on the consistent brilliance of their live performances reciting precomposed music[3]. More recently, questions in computer music have found themselves inexorably tied up with the booming interest in improvised music, possibly for obvious reasons of suitability – what better test of musicianship than the coming together of the live and the compositional? – but possibly also due to trends in music that are politically broader and more profound, such as discussed by Lewis (2002).

From the point of view of analysing computer musicianship, therefore, it seems more appropriate to state that there is no difference between what is live and what is composed; both can be regarded as performances and neither can be judged for their technological merit from a single instance[4]. And whilst the *humanness* of human musical performance is clearly highly regarded by most people, it is important to acknowledge the possibility that this is not because human musical performance is indelibly written into human music perception, but because we are, for obvious reasons, most familiar with human musical performance, and, as Rowe says, there is presently nothing that approximates it. As is widely observed, individual musical tastes vary to the point of mutual exclusion, and musical styles follow a temporal dynamic that is so rich that anyone should doubt the rigidity of musical tolerance to stop at computer composed music. Meanwhile, socio-cultural factors reify the importance of *human* performance in music: even in studio produced music, such as dance music played by a DJ, the visible act of performance is relished, and the relationship between the audience and that performer is viewed as critical. This highlights the strong relationship between spectacle and musical production, but also a key distinction. Perhaps popular music will always need human performative elements, but this apparently does not place a particularly great constraint on how the music is actually produced.

The discussion of modes of musical production and musical style in this section is aimed at drawing attention to the generally isolated and individualist use of most computer music systems, including alife music systems. In the previous section we argued that individualistic approaches to alife music do not sit well with the understanding that the autonomy of real living systems is contingent on the flexibility with which nature provides what Gibson dubbed *affordances*. This leads to the implication that new approaches to musical production and style are as crucial to the development of concepts of alife in music as are direct advances in the field of alife and more literal developments in applying alife to music in individual situations.

5 How Can We Do Strong Alife Music?

The above discussion would be heading for a completely negative conclusion if it wasn’t for the fact that our technological environment is changing the way that people make music, as well as aspects of our social organisation. Most importantly for a notion of alife music, as music producers and consumers increase the degree to which they create music in networked environments, they increasingly contribute to an environment which is genuinely rich in its capacity to generate affordances valuable to evolving software systems. Musicians making music on computers connected to the internet allow for music software that shares information about these various musical contexts. By linking up musical contexts in this way it is possible to see beyond the individualist limitations discussed above, through the creation of a rich and diverse environment. Then no single individual need determine the fate of an alife system, and as a direct consequence of

[2]We say this with some caution because a lot of performance may be associated with the music in clubs or on music videos, and the music may sample other musical performances, and thus disguise human performative involvement in its otherwise mechanical production. Despite this we hold that there are examples of purely non-performative dance music production, as well as electroacoustic and computer tape music.

[3]Sometimes because they achieve an almost superhuman consistency in their performances, which would make for an ironic criterion for the evaluation of computer performers!

[4]One might argue that an exception lies in cases where, for example, a computer system evaluates a whole piece in order to propose a modification to that piece. This is a process that cannot be placed in a live context because it would require knowledge of the future, suggesting a fundamental difference between live and compositional contexts. But whilst this difference does indeed exist, such cases do not undermine the assertion that computer composition is essentially performative.

this, alife systems need not be subject to any one single functional expectation or interpretation.

Such an observation is not an original contribution by this author. Amongst the various commentators who have discussed the possibility of networked communities of users interacting with communities of software agents, the most significant effort has been made by the Hybrid Society (HS) project (Romero et al., 2003). The HS project aims to explore approaches to artificial evolution in a rich world of interaction generated from a group of individuals interacting over a network, including the internet. They point to the problem of *fatigue* associated with a single user IGA approach, proposing that a multi-user approach is a potential solution to this problem. The HS environment does not make any specific demands about what its agents do, and how they are understood by its human users, except to define a general purpose interaction paradigm. Thus the HS environment provides an appropriate framework for a strong alife music to develop, or at least for interesting provisional research in this domain to take place. However, it does explicitly require that software agents and human users be viewed as equivalent and equal actors in the network, both have the ultimate goal of producing aesthetic artworks.

Although non academic enthusiasts are invited to participate in experiments over the internet, the HS project is strictly executed in an academic experimental manner; participants get involved out of academic interest. This is hardly a surprising state of affairs for current alife music practice, but it is one that tightly limits the potential user-base of any such system, and maintains the separation between real artists working in the real world and the environment in which they interact when they turn their attention to the HS project. For strong alife music the network of potentially interested participants needs to not be restricted by this constraint, and to diversify to the extent of the diversity of current music practice. Also, in order to further consolidate the valuable differences between alife and AI, it is important not to conflate alife agents with humans. A master-pet relationship is a more fitting analogy than one of equivalence.

To expand the user base of alife music software two things need to happen. Firstly, regular music software needs to *go alife*. That is, in normal musical contexts certain elements should be recontextualised as adaptive agents and should be able to gather data that informs the design of new systems, that possibly replace old systems. The most important first step to this is that it continues to behave like regular software. Secondly, music makers need to open up to software that has erratic, unpredictable, idiosyncratic behaviour. The crux of this paper is that a common practice alife music that would satisfy this second condition is feasible, already heavily active, but contingent on the existence of music alife software for its development. It is not necessary to try to define how this common practice alife music would work, but we can consider some questions about it. For this purpose we juxtapose two sci-fi vignettes that capture the essential differences between an imagined alife music and more traditional views of computer intelligence in music:

> *Sci-fi scenario 1:* John is at the concert hall setting up for his rehearsal. He opens two violin cases and a box containing the Z7 concert-grade violin recital robot. He sets up the Z7's shoulder and arm mechanism on the stage and mounts one of the violins on it. He plugs in the Z7's hardware controller to the shoulder and arm mechanism, and also plugs in a microphone which he points towards himself. He powers up the hardware. A light turns on, red at first, then green after a couple of seconds. The Z7 sounds a pre-recorded note, and they begin tuning their violins…
>
> *Sci-fi scenario 2:* Mark has just got home from school. He logs into his PC, connects his electric guitar to the sound card and starts up AudioLife 5.2. The program asks him whether he would like to load an existing environment from his local machine, or search online for active environments. He choses to go online, and the software provides a list of current active environments. He browses by category, finally settling for CragFunk, and picks an environment at random from the list. The program asks him if he would like to chose any MIDI or audio files as source material for the software agents, the alternative being that they generate their own material from scratch…

The *meaning of life* of the alife music system in the second scenario does not come from the fact that it is *performing live* in a context that is accepted as a site of *real* music. A formal performance (strictly scored or completely improvised) is the tip of the iceberg of a rich inhabitable musical environment. Such a context can be understood to establish a smokescreen between the audience and the performance, and generally imposes strict limitations on the performer's activity (even in the improvised context). Rather it comes from the exploratory day-to-day interactions between human and musical system, which has to be interactive and exploratory for there to be any meaning to the system's action. Thus alife music systems might manifest themselves as small components of existing software systems such as VST plug-ins, plug-ins to music playback software such as Apple's iTunes, and as as objects in computer music environments such as Max/MSP (www.cycling74.com), PD (www.puredata.info) and SuperCollider (www.audiosynth.com).

Consider the domain of this exploratory day-to-day musical interaction. How does it differ from context to context? In the vignette the context is that of a child outside of his normal educational routine engaging with some kind of contemporary urban music. He is not a professional. He does not necessarily know what he wants to get out of this interaction, like most children he has a limited sense of what is possible musically, and having grown up with this kind of alife software commonplace he unthinkingly accepts its legitimacy.

In this context there is an important opportunity for the alife system to vary. Each time a download is made from the list it may be the mutated or crossbred offspring of earlier successful agents. This is different from the variation of a system that is designed to be creative; the user is not the designer of the system, and he does not require that the system is creative and therefore variable, even if he *would* ultimately like the system to pass through long-term changes. The system need not change at all once he has downloaded it. What is important is that when the user does want something to be different they go about finding it in an alife way. This may mean asking for a new variation, as in an IGA, but it could also involve manually tweaking the system as long as the information from this interaction could be interpreted as a significant interaction with its environment, and used to inform later evolution.

There are numerous significant implementation questions surrounding the kind of system that would fulfil this goal. Our concern is only with the context in which this could happen. Could an online multiuser evolving system fit easily with the goals and desires of the people using the system? It would be problematic if users stuck with behaviours they liked and never look for new ones, or if they became frustrated with constantly searching for behaviours which acted in ways they did not understand rather than designing behaviours from known methods. They would soon go back to tried and tested music making, and the system would freeze to a halt. Likewise, as in any evolutionary computing approach, the stagnation of the system itself in local optima is a constant threat to the development of genuinely interesting behaviours, such as the qualities of good alife music systems discussed at the beginning of section 2. Overcoming these obstacles would be an important development in strong alife music, and the design of such systems would ideally be gentle on their demands from end users, or somehow seductive. However, it is interesting to consider the many forms of artistic practice (possibly only in recent history) that depend more on our editorship of existing systems than on a thoroughly creative act. DJing and remixing activities, extending to musical genres such as bootleg and mashup, epitomise this approach to creativity. This is clearly a new cultural paradigm, but it may also be a more explicit expression of what creativity has always been about (*c.f.*, Koestler, 1967; Boden, 1990; Csikszentmihalyi, 1990, 1999). Indeed, various views on the creative process focus on the process of *generate and test*. Thus an optimistic view of strong alife music is that it is actually perfectly suited to our collective creative activities, and blurs the perceived boundaries between the individual as creative system, the society as creative system and software as creative system.

6 Conclusion

Many of the professional music producers of today more often than not learnt their skills outside of the classroom and in the *bedroom studio*, a whole music practice emergent on the technology that was designed around other existing music practices of that time. In this paper we have asked how the strange marriage of artificial life and music *could* come to take on a meaning and significance that truly bears on the principles of alife. We have alluded to the emergence of a new social context that is entire fantasy but with the simple goal of thinking through the possible ways in which a technological aim and a social practice may come together. It would be wrong to assume that by highlighting this context as a possibility, no matter how theoretically correct it may be, it would be simple to evoke it through some kind of social engineering. In the above discussion we rely on the notion that creative individuals find new uses for existing technology, and there is no reason the believe that the uses they find for alife music systems would pay any homage to the principles of alife. All the same, it is exciting to consider the results of designing multiuser evolutionary systems for popular use that are based on the principle of providing a rich variable evolutionary environment and to study ways in which these systems are taken up, in which case it is vital to acknowledge the role of social trends in the success or failure of such systems, as well as their actual design. This suggests an interesting new direction for alife-based music informatics, which would need to incorporate the analysis of collective human social behaviour in its remit. It also suggests new approaches to the study of computational creativity, in which we sever questions of creativity from intelligence – artistic creativity becomes analogous to the creativity of nature – as well as from the individual – individuals act creatively, but this is only one layer of a greater collective creative process.

Acknowledgements

Oliver Bown's research is supported by a bursary from the Department of Computing, Goldsmiths College. We would like to thank Alice Eldridge for valuable discussions and the three anonymous reviewers for their comments.

References

Beer, R. (1996). Toward the evolution of dynamical neural networks for minimally cognitive behavior. In *From animals to animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, pages 421–429. MIT Press.

Berry, R. and Dahlstedt, P. (2003). Artificial life: Why should musicians bother? *Contemporary Music Review*, 22(3):57–67.

Blackwell, T. and Young, M. (2004). Self-organised music. *Organised Sound*, 9(2):137–150.

Boden, M. (1990). *The Creative Mind.* George Weidenfeld and Nicholson Ltd.

Brooks, R. A. (1990). Elephants don't play chess. *Robotics and Autonomous Systems*, 6:3–15.

Collins, N. (2006). *Towards Autonomous Agents for Live Computer Music: Realtime Machine Listening and Interactive Music Systems.* PhD thesis, Centre for Science and Music, Faculty of Music, University of Cambridge.

Csikszentmihalyi, M. (1990). The domain of creativity. In Runco, M. and Albert, R. S., editors, *Theories of Creativity.* Sage Publications.

Csikszentmihalyi, M. (1999). Implications of a systems perspective for the study of creativity. In Sternberg, R. J., editor, *The Handbook of Creativity*. CUP.

Eno, B. (1996). *A Year With Swollen Appendices*. Faber and Faber.

Koestler, A. (1967). *The Ghost in the Machine*. Hutchinson and Co.

Lewis, G. E. (2002). Improvised music after 1950: Afrological and eurological perspectives. *Black Music Research Journal*, 22:215–246.

Lovelock, J. (1979). *Gaia. A New Look at Life on Earth*. OUP.

Miranda, E. (2001). *Composing Music with Computers*. Focal Press.

Owings, D. H. and Morton, E. S. (1998). *Animal Vocal Communication: A New Approach*. Cambridge University Press.

Romero, J., Machado, P., Santos, A., and Cardoso, A. (2003). On the development of critics in evolutionary computation artists. In *Applications of Evolutionary Computing: EvoWorkshops 2003: EvoBIO, EvoCOP, EvoIASP, EvoMUSART, EvoROB, and EvoSTIM, Essex, UK, April 14-16, 2003. Proceedings*, volume 2611/2003 of *Lecture Notes in Computer Science*, pages 559–569.

Rowe, R. (1993). *Interactive Music Systems*. MIT Press.

Slocum, A., Downey, D., and Beer, R. (2000). Further experiments in the evolution of minimally cognitive behavior: From perceiving affordances to selective attention. In Meyer, J., Berthoz, A., Floreano, D., Roitblat, H., and Wilson, S., editors, *From Animals to Animats 6: Proceedings of the Sixth International Conference on Simulation of Adaptive Behavior*, pages 430–439. MIT Press.

Evaluating Cognitive Models of Musical Composition

Marcus T. Pearce and Geraint A. Wiggins
Centre for Cognition, Computation and Culture
Goldsmiths, University of London
New Cross, London SE14 5SG, UK
{m.pearce,g.wiggins}@gold.ac.uk

Abstract

We present a method for the evaluation of creative systems. We deploy a learning-based perceptual model of musical melodic listening in the generation of tonal melodies and evaluate its output quantitatively and objectively, using human judges. Then we show how the system can be enhanced by the application of mathematical methods over data supplied by the judges. The outcome to some extent addresses the criticisms of the experts. We suggest that this is a first step on the road to autonomously learning, introspective, creative systems.

1 Introduction

We examine, at the computational level, the demands of the melodic composition task, focusing on constraints placed on the representational primitives and the expressive power of the composition system. We use three multiple-feature Markov models trained on a corpus of chorale melodies to generate novel pitch structures for seven existing chorale melodies. We propose null hypotheses that each model is consistently capable of generating chorale melodies that are rated as equally successful examples of the style as the original chorale melodies in our dataset. To examine the hypotheses, experienced judges rated the generated melodies together with the original chorale melodies, using a variant of the Consensual Assessment Technique (Amabile, 1996) for investigating psychological components of human creativity. The results warrant rejection of the null hypothesis for all three of the systems. Even so, further analysis identifies some objective features of the chorale melodies that exhibit significant relationships with the ratings of stylistic success, suggesting how the computational models fail to meet intrinsic stylistic constraints of the genre. Adding new features to address these concerns significantly improves our systems' prediction performance.

We present our experiment and the evaluation method, which, we suggest, forms a basis for systems capable of introspection based on feedback on their output.

2 Background

2.1 Music Generation from Statistical Models

Conklin (2003) examines four methods of generating high-probability music according to a statistical model. The simplest is sequential random sampling: an event is sampled from the estimated event distribution at each sequential position up to a given length. Events are generated in a random walk, so there is a danger of straying into local minima in the space of possible compositions. Even so, most statistical generation of music uses this method.

The Hidden Markov Model (HMM) addresses these problems; it generates observed events from hidden states (Rabiner, 1989). An HMM is trained by adjusting the probabilities conditioning the initial hidden state, the transitions between hidden states and the emission of observed events from hidden states, so as to maximise the probability of a training set of observed sequences. A trained HMM can be used to estimate the probability of an observed sequence of events and to find the most probable sequence of hidden states given an observed sequence of events. This can be achieved efficiently for a first-order HMM using the Viterbi algorithm; a similar algorithm exists for first-order (visible) Markov models. However, Viterbi's time complexity is exponential in the context length of the underlying Markov model (Conklin, 2003).

Tractable methods for sampling from complex statistical models (such as those presented here) which address the limitations of random sampling do exist, however (Conklin, 2003). The *Metropolis-Hastings algorithm* is a Markov Chain Monte Carlo (MCMC) sampling method (MacKay, 1998). The following description applies it within our generation framework. Given a trained multiple-feature model m for some basic feature τ_b, in order to sample from the target distribution $p_m(s \in [\tau_b]^*)$, the algorithm constructs a Markov chain in the space of possible feature sequences $[\tau_b]^*$ as follows:

1. number of iterations $N \leftarrow$ a large value; iteration number $k \leftarrow 0$; initial state $s_0 \leftarrow$ some feature sequence $t_1^j \in [\tau_b]^*$ of length j;
2. select event index $1 \leq i \leq j$ at random or based on

some ordering of the indices;

3. let s'_k be the sequence obtained by replacing event t_i at index i of s_k with a new event t'_i sampled from a distribution q which may depend on the current state s_k – in the present context, an obvious choice for q would be $\{p_m(t|t_1^{i-1})\}_{t\in[\tau_b]}$;
4. accept the proposed sequence with probability
$$\min\left[1,\ \frac{p_m(s'_k)\ \cdot\ q(t_i)}{p_m(s_k)\ \cdot\ q(t'_i)}\right];$$
5. if accepted, $s_{k+1} \leftarrow s'_k$, else $s_{k+1} \leftarrow s_k$;
6. if $k < N$, k++ and iterate from 2, else return s_k.

If N is large enough, the resulting event sequence s_{N-1} is guaranteed to be an unbiased sample from the target distribution $p_m([\tau_b]^*)$. However, there is no method of assessing the convergence of MCMCs nor of estimating the number of iterations required to obtain an unbiased sample (MacKay, 1998). Because these sampling algorithms explore the state space using a random walk, they can still be trapped in local minima.

Event-wise substitution is unlikely to provide a satisfactory model of phrase- or motif-level structure. Our model has a short-term component, to model intra-opus structure, but generation still relies on single-event substitutions. Pattern-discovery algorithms may be used to reveal phrase level structure, which may subsequently be preserved during stochastic sampling (Conklin, 2003).

2.2 Evaluating Computer Models of Composition

Analysis by synthesis evaluates computational models of composition by generating pieces and evaluating them with respect to the objectives of the implemented model. The method has a long history; Ames and Domino (1992) argue that a primary advantage of computational analysis of musical style is the ability to evaluate new pieces generated from an implemented theory. However, evaluation of the generated music raises methodological issues which have typically compromised the potential benefits thus afforded (Pearce et al., 2002). Often, compositions are evaluated with a single subjective comment, *e.g.,*: "[the compositions] are realistic enough that an unknowing listener cannot discern their artificial origin" (Ames and Domino, 1992, p. 186). This lack of precision makes it hard to compare theories intersubjectively.

Other research has used expert stylistic analyses to evaluate computer compositions. This is possible when a computational model is developed to account for some reasonably well-defined stylistic competence or according to critical criteria derived from music theory or music psychology. For example, Ponsford et al. (1999) gave an informal stylistic appraisal of the harmonic progressions generated by their n-gram models.

However, even when stylistic analyses are undertaken by groups of experts, the results obtained are typically still qualitative. For fully intersubjective analysis by synthesis, the evaluation of the generated compositions must be empirical. One could use an adaptation of the Turing test, where subjects are presented with pairs of compositions (one computer-generated, the other human-composed) and asked which they believe to be the computer-generated one (Marsden, 2000). Musical Turing tests yield empirical, quantitative results which may be appraised intersubjectively. They have demonstrated the inability of subjects to distinguish reliably between computer- and human-composed music. But the method can be biased by preconceptions about computer music, allows ill-informed judgements, and fails to examine the criteria being used to judge the compositions.

2.3 Evaluating Human Composition

Amabile (1996) proposes a conceptual definition of creativity in terms of processes resulting in novel, appropriate solutions to heuristic, open-ended or ill-defined tasks. However, while agreeing that creativity can only be assessed through subjective assessments of products, she criticises the use of *a priori* theoretical definitions of creativity in rating schemes and failure to distinguish creativity from other constructs. While a conceptual definition is important for guiding empirical research, a clear operational definition is necessary for the development of useful empirical methods of assessment. Accordingly, she presents a consensual definition of creativity in which a product is deemed creative to the extent that observers who are familiar with the relevant domain independently agree that it is creative. To the extent that this construct is internally consistent (independent judges agree in their ratings of creativity), one can empirically examine the objective or subjective features of creative products which contribute to their perceived creativity.

Amabile (1996) used this operational definition to develop the *consensual assessment technique* (CAT), an empirical method for evaluating creativity. Its requirements are that the task be open-ended enough to permit considerable flexibility and novelty in the response, which must be an observable product which can be rated by judges. Regarding the procedure, the judges must:

1. be experienced in the relevant domain;
2. make independent assessments;
3. assess other aspects of the products such as technical accomplishment, aesthetic appeal or originality;
4. make relative judgements of each product in relation to the rest of the stimuli;
5. be presented with stimuli and provide ratings in orders randomised differently for each judge.

Most importantly, in analysing the collected data, the inter-judge reliability of the subjective rating scales must be determined. If—and only if—reliability is high, we may correlate creativity ratings with other objective or subjective features of creative products.

Numerous studies of verbal, artistic and problem solving creativity have demonstrated the ability of the CAT to obtain reliable subjective assessments of creativity in a range of domains (Amabile, 1996, ch. 3, gives a review).

The CAT overcomes the limitations of the Turing test in evaluating computational models of musical composition. First, it requires the use of judges expert in the task

System	Features	H
A	`Pitch`	2.337
B	`Int1stInPiece`, `ScaleDegree` ⊗`DurRatio`, `Thread1stInPhrase`	2.163
C	`Interval`⊗`Duration`, `ScaleDegree` ⊗`Int1stInPiece`, `Pitch`⊗`Duration`, `ScaleDegree`⊗`1stInBar`, `ThreadTactus`, `ScaleDegree`⊗`Duration`, `Interval`⊗`DurRatio`, `Int1stInPiece`, `Thread1stInPhrase`	1.953

Table 1: The component features of Systems A, B and C and their average information content computed by 10-fold cross-validation over the dataset.

domain. Second, since it has been developed for research on human creativity, no mention is made of the computational origins of the stimuli; this avoids bias due to preconceptions. Third, and most importantly, the methodology allows more detailed examination of the objective and subjective dimensions of the creative products. Crucially, the objective attributes of the products may include features of the generative models (corresponding with cognitive or stylistic hypotheses) which produced them. Thus, we can empirically compare different musicological theories of a given style or hypotheses about the cognitive processes involved in composing in that style.

3 The Experiment

3.1 Introduction

Following Johnson-Laird (1991), we analyse the computational constraints of the melody composition task in two ways: first, examining whether our learned finite context grammars can compose stylistically-successful melodies or whether more expressive grammars are needed; and second, determining which representational structures are needed for the composition of successful melodies.

Our experiment is designed to test the hypothesis that our statistical models are capable of generating melodies which are deemed stylistically successful in the context of a specified tradition. Three multiple-feature Markov models (Pearce, 2005) trained on a dataset of chorale melodies were used to generate melodies which were then empirically evaluated: System A is a single-feature system; System B is a multiple-feature system developed through forward, stepwise feature selection to provide the closest fit to the human expectancy judgements obtained by Manzara et al. (1992); and System C is a multiple-feature system developed through forward, stepwise feature selection to yield the best prediction performance over the chorale dataset. The Systems were parameterised optimally and differ only in the features they use (Table 1).

Our work differs in several ways from extant statistical modelling for music generation, in particular, in that no symbolic constraints were imposed on the generation process—it was based entirely on the learned models. This focuses the analysis more sharply on the inherent capacities of statistical finite context grammars, since our goal was to examine the synthetic capabilities of purely statistical, data-driven models of melodic structure.

Our strategy improves on previous work in several ways. The variable order selection policy of PPM* (Cleary and Teahan, 1997) is used to address concerns that low, fixed order models tend to generate features uncharacteristic of the target style (Ponsford et al., 1999). Other model parameters are optimised to improve prediction performance over a range of different melodic styles. Systems B and C operate over rich representational spaces supplied by the multiple-feature framework; their features were selected on the basis of objective and empirical criteria (*cf.* Conklin and Witten, 1995). Our Systems use a novel model combination strategy, which improves prediction performance over the chorale dataset (Pearce, 2005). While most previous approaches used sequential random sampling to generate music from statistical models, in the present research melodies were generated using Metropolis sampling. We expect that this method will be capable of generating melodies which are more representative of the inherent capacities of the Systems. We do not propose Metropolis sampling as a cognitive model of melodic composition, but use it merely as a means of generating melodies which reflect the internal state of knowledge and capacities of the trained models.

Finally, to evaluate the systems as computational models of melodic composition, we developed a method based on the CAT. The method, described fully by Pearce (2005), obtains ratings by expert judges of the stylistic success of computer generated compositions and existing compositions in the target genre. The empirical nature of this method makes it preferable to the exclusively qualitative analyses typically adopted and we expect it to yield more revealing results than the Turing test methodology.

3.2 Hypotheses

We use three different Systems to examine which representational structures are needed for competent melody generation. Our null hypotheses are that each System can generate melodies rated as equally stylistically successful in the target style as existing, human-composed melodies. We expect the null hypothesis for the simplistic System A to be refuted.

For System B, Baroni's (1999) proposal that composition and listening involve equivalent grammatical structures is relevant. If the representational structures underlying perception and composition of music are similar, we would expect grammars which model perceptual processes well to generate satisfactory compositions. Since System B represents a satisfactory model of the perception of pitch structure in the chorale genre, we may expect to retain the null hypothesis for this system.

Pearce and Wiggins (2006) demonstrate a relationship between prediction performance and fit to human expectancy data (Manzara et al., 1992), suggesting that human perceptual systems base their predictions on uncertainty-reducing representational features. In terms of model selection for music generation, highly predictive theories of a musical style, as measured by information content, should generate original and acceptable works in the style (Conklin and Witten, 1995). Systems A, B and C in turn exhibit decreasing uncertainty in predicting unseen melodies from the dataset (Table 1). Therefore, we may expect to retain the null hypothesis for System C.

3.3 Method

3.3.1 Judges

Our judges were 16 music researchers or students at City University, London, Goldsmiths, University of London, and the Royal College of Music. Five were male and eleven female, and their age range was 20–46 years (mean 25.9, SD 6.5). They had been formally musically trained for 2–40 years (mean 13.8, SD 9.4). Seven judges reported high familiarity with the chorale genre and nine were moderately familiar. All judges received a nominal payment, and worked for approximately an hour.

3.3.2 Apparatus and Stimulus Materials

Our dataset is a subset of the chorale melodies placed in the soprano voice and harmonised in four parts by J. S. Bach. These melodies are characterised by stepwise patterns of conjunct intervallic motion and simple, uniform rhythmic and metric structure. Phrase structure is explicitly notated. Most phrases begin on the tonic, mediant or dominant and end on the tonic or dominant; the final phrase almost always ends with a cadence to the tonic.

Our stimuli were as follows. Seven existing *base* melodies were randomly selected from the set of chorales in the midrange of the distribution of average information content (cross-entropy) values computed by System A. All 7 were in common time; 6 were in major keys and 1 was minor; they were 8–14 bars (mean 11.14) and 33–57 events (mean 43.43) long. The base melodies were removed from the training dataset.

7 novel melodies were generated by each System, *via* 5000 iterations of Metropolis sampling using the 7 base chorales as initial states. Only pitch was sampled: time and key signatures and rhythmic and phrase structure were left unchanged. Figure 1 shows one base chorale melody and the three melodies generated using it; Pearce (2005) gives further examples.

Each melody was stored as a quantised MIDI file. A pattern of velocity accents was added to emphasise the metrical structure and a one-beat rest was inserted after each fermata to disambiguate the phrase structure. The stimuli were recorded to CD-quality audio files on a PC using the piano tone of a Roland XP10 synthesiser connected via MIDI to a Terratec EWS88 MT soundcard, at a uniform 90 beats per minute. They were presented over Technics RP-F290 stereo headphones fed from a laptop PC running a software media player. The judges recorded their responses in writing in a response booklet.

3.3.3 Procedure

Our judges supplied their responses individually and received instructions verbally and in writing. We told them they would hear a series of chorale melodies in the style of Lutheran hymns and asked them to listen to each entire melody before answering two questions about it by placing circles on discrete scales in the response booklet. The

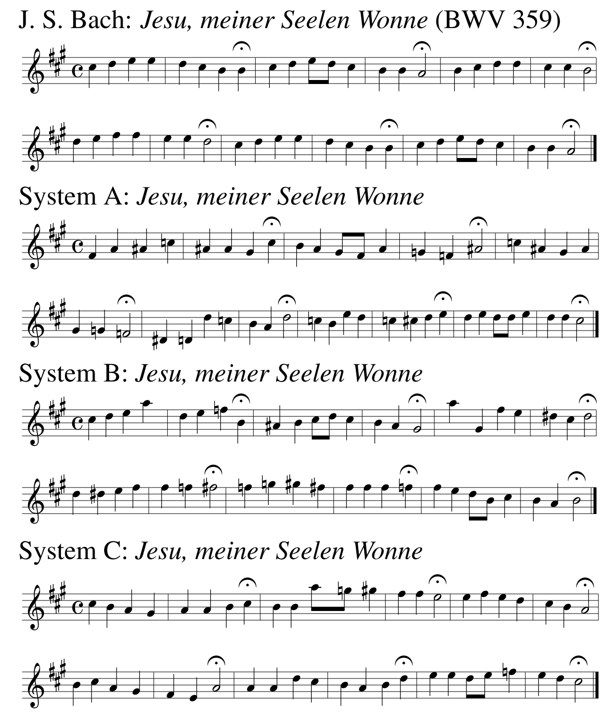

J. S. Bach: *Jesu, meiner Seelen Wonne* (BWV 359)
System A: *Jesu, meiner Seelen Wonne*
System B: *Jesu, meiner Seelen Wonne*
System C: *Jesu, meiner Seelen Wonne*

Figure 1: An example of one base chorale melody and the three melodies generated using it.

first question[1] was, "How successful is the composition as a chorale melody?" Judges were advised that their answers should reflect such factors as conformity to important stylistic features, tonal organisation, melodic shape and interval structure; and melodic form. Answers to this question were given on a seven-point numerical scale, 1–7, with anchors marked low (1), medium (4) and high (7). To promote an analytic approach to the task, judges were asked to briefly justify their responses to the first question. The second question was, "Do you recognise the melody?" Judges were advised to answer "yes" only if they could specifically identify the composition as one they were familiar with.

We explained to the judges that after both questions had been answered for a melody, they could listen to the next one by pressing a single key on the PC. We asked them to bear in mind that their task was to rate the composition of each melody rather than the performance and urged them to use the full range of the scales, reserving 1 and 7 for extreme cases. There were no constraints on the time taken to answer the questions.

The experiment began with a practice session during which judges heard two melodies from the same genre (but not one of those in the test set). These practice trials were intended to set a judgemental standard for the subsequent test session. This departs from the CAT, which encourages judges to rate each stimulus in relation to the others by experiencing all stimuli before making their ratings. However, here, we intended the judges to use their expertise to rate the stimuli against an absolute standard: the body of existing chorale melodies. Judges responded

[1]This is a variant on the original CAT, whose primary judgement was about creativity. We justify this on the grounds that stylistic success is a directly comparable kind of property.

as described above for both of the items in the practice block. The experimenter remained in the room for the duration of the practice session after which the judges were given an opportunity to ask any further questions; he then left the room before the start of the test session.

In the test session, the 28 melodies were presented to the judges, who responded to the questions. The melodies were presented in random order subject to the constraints that no melody generated by the same system nor based on the same chorale were presented sequentially. A reverse counterbalanced design was used, with eight of the judges listening to the melodies in one such order and the other eight listening to them in the reverse order.

After the test session, the judges filled out a questionnaire detailing their age, sex, number of years of music training (instrument and theory) and familiarity with the chorales harmonised by J. S. Bach (high/medium/low).

3.4 Results

3.4.1 Inter-judge Consistency

We report analyses of the 28 melodies from our test session: we discarded the data from the practice block. First, we examine the consistency of the judges' ratings.

All but two of the 120 pairwise correlations between judges were significant at $p < 0.05$ with a mean coefficient of $r(26) = 0.65$ ($p < 0.01$). Since there was no apparent reason to reject the judges involved in the two non-significant correlations, we did not do so. This high consistency warrants averaging the ratings for each stimulus across individual judges in subsequent analyses.

3.4.2 Presentation Order and Prior Familiarity

Two factors which might influence the judges' ratings are the order of presentation of the stimuli and prior familiarity. The correlation between the mean success ratings for judges in the two groups was $r(26) = 0.91, p < 0.01$ indicating a high degree of consistency across the two orders of presentation, and warranting the averaging of responses across the two groups; and, although the mean success ratings tended to be slightly higher when judges recognised the stimulus, a paired t test revealed no significant difference: $t(6) = 2.07, p = 0.08$.

3.4.3 Influence of Generative System and Base Chorale

Now we examine the primary question: the influence of generative system on the ratings of stylistic success. The mean success ratings for each stimulus are shown in Table 2. The mean ratings suggest that the original chorale melodies were rated higher than the computer-generated melodies while the ratings for the latter show an influence of base chorale but not of generative system. Melody C249 is an exception, attracting high average ratings of success. Our preferred analysis would have been a multivariate ANOVA using within-subjects factors for generative system with 4 levels (Original, System A, B, C) and base chorale with 7 levels (249, 238, 365, 264, 44, 153 and 147) with the null hypotheses of no main or interaction effects of generative system or base chorale. However, Levene's test revealed significant non-homogeneity of variance with respect to the factor for generative system

Base	System A	System B	System C	Original	Mean
249	2.56	2.44	5.00	6.44	4.11
238	3.31	2.94	3.19	5.31	3.69
365	2.69	1.69	2.50	6.25	3.28
264	1.75	2.00	2.38	6.00	3.03
44	4.25	4.38	4.00	6.12	4.69
141	3.38	2.12	3.19	5.50	3.55
147	2.38	1.88	1.94	6.50	3.17
Mean	2.90	2.49	3.17	6.02	3.65

Table 2: The mean success ratings for each stimulus and means aggregated by generative system and base chorale.

Statistic	System A	System B	System C	Original
Median	2.86	2.57	3.07	5.93
Q1	2.68	2.25	2.68	5.86
Q3	3.29	2.75	3.61	6.29
IQR	0.61	0.50	0.93	0.43

Table 3: The median, quartiles and inter-quartile range of the mean success ratings for each generative system.

$F(3) = 6.58, p < 0.01$, so ANOVA was not applicable. Therefore, we used Friedman's rank sum tests, as a non-parametric alternative; this does not allow examination of interactions between the two factors.

We examined the influence of generative system in an unreplicated complete blocked design using the mean success ratings aggregated for each subject and generative system across the individual base chorales. Summary statistics for this data are shown in Table 3. The Friedman test revealed a significant within-subject effect of generative system on the mean success ratings: $\chi^2(3) = 33.4, p < 0.01$. We compared the factor levels pairwise using Wilcoxon rank sum tests with Holm's Bonferroni correction for multiple comparisons: the ratings for the original chorale melodies differ significantly from the ratings of melodies generated by all three computational systems ($p < 0.01$). Furthermore, the mean success ratings for the melodies generated by System B were found to be significantly different from those of the melodies generated by Systems A and C ($p < 0.03$). These results suggest that none of the systems is capable of consistently generating chorale melodies which are rated as equally stylistically successful as those in the dataset and that System B performed especially poorly.

4 Learning from Qualitative Feedback

4.1 Objective Features of the Chorales

Next, we aim to explain how the Systems lack compositionally, by examining which objective musical features of the stimuli the judges used in making their ratings of stylistic success. This could explain how the systems are lacking compositionally. To achieve this, we analysed the stimuli qualitatively and developed a set of corresponding objective descriptors, which we then applied in a series of multiple regression analyses using the rating scheme, averaged across stimuli, as a dependent variable. We now present the descriptive variables, their quantitative coding and the analysis results.

The chorales generated by our systems are mostly not very stylistically characteristic of the dataset, especially in higher-level form. From the judges' qualitative comments, we identified stylistic constraints describing the stimuli and distinguishing the original melodies. We grouped them into five categories—pitch range; melodic structure; tonal structure; phrase structure; and rhythmic structure—each covered by a predictor variable.

Pitch Range The dataset melodies span a pitch range of about an octave above and below C_5, favouring the centre of this range. The generated melodies are constrained to this range, but some tend towards extreme tessitura. We developed a predictor variable *pitch centre* to capture this difference, reflecting the absolute distance, in semitones, of the mean pitch of a melody from the mean pitch of the dataset (von Hippel, 2000). Another issue is the overall pitch range of the generated chorales. The dataset melodies span an average range of 11.8 semitones. By contrast, several of the generated melodies span pitch ranges of 16 or 17 semitones, with a mean pitch range of 13.9 semitones; others have a rather narrow pitch range. We captured these qualitative considerations in a quantitative predictor variable *pitch range*, representing the absolute distance, in semitones, of the pitch range of a melody from the mean pitch range of the dataset.

Melodic Structure There are several ways in which the generated melodies do not consistently reproduce salient melodic features of the original chorales. The most obvious is a failure to maintain a stepwise pattern of movement. While some generated melodies are relatively coherent, others contain stylistically uncharacteristic leaps of an octave or more. Of 9042 intervals in the dataset melodies, only 57 exceed a perfect fifth and none exceeds an octave. To capture these deviations, we created a quantitative predictor variable called *interval size*, representing the number of intervals greater than a perfect octave in a melody. The generated chorales also contain uncharacteristic discords such as tritones or sevenths. Only 8 of the 9042 intervals in the dataset are tritones or sevenths (or their enharmonic equivalents). To capture these deviations, we created a quantitative predictor variable *interval dissonance*, representing the number of dissonant intervals greater than a perfect fourth in a melody.

Tonal Structure Since System A operates exclusively over representations of pitch, it is not surprising that most of its melodies fail to establish a key note and exhibit little tonal structure. However, we might expect Systems B and C to do better. While the comments of the judges suggest otherwsie, they may have arrived at a tonal interpretation at odds with the intended key of the base chorale. To independently estimate the perceived tonality of the test melodies, Krumhansl's (1990) key-finding algorithm, using the revised key profiles of Temperley (1999) was applied to each of the stimuli. The algorithm assigns the correct keys to all seven original chorale melodies. While the suggested keys of the melodies generated by System A confirm that it does not consider tonal constraints, the melodies generated by Systems B and C retain the key of their base chorale in two and five cases respectively. Furthermore, especially in the case of System C, deviations from the base chorale key tend to be to related keys (either in the circle of fifths or through relative and parallel major/minor relationships). This suggests some success on the part of the more sophisticated systems in retaining the tonal characteristics of the base chorales.

Nonetheless, the generated melodies are often unacceptably chromatic, which obscures the tonality. Therefore, we developed a quantitative predictor called *chromaticism*, representing the number of chromatic tones in the algorithm's suggested key.

Phrase Structure The generated chorales typically fail to reproduce the implied harmonic rhythm of the originals and its characteristically strong relationship to phrase structure. In particular, while some of the generated melodies close on the tonic, many fail to imply stylistically satisfactory harmonic closure. To capture such effects, we created a variable called *harmonic closure*, which is 0 if a melody closes on the tonic of the key assigned by the algorithm and 1 otherwise. Secondly, the generated melodies frequently fail to respect thematic repetition and development of melodic material embedded in the phrase structure of the chorales. However, these kinds of repetition and development of melodic material are not represented in the present model. Instead, as a simple indicator of complexity in phrase structure, we created a variable *phrase length*, which is 0 if all phrases are of equal length and 1 otherwise.

Rhythmic Structure Although the chorale melodies in the dataset tend to be rhythmically simple, the judges' comments revealed that they were taking account of rhythmic structure. Therefore, we adapted three further quantitative predictors modelling rhythmic features from Eerola and North's (2000) expectancy-based model of melodic complexity. *Rhythmic density* is the mean number of events per tactus beat. *Rhythmic variability* is the degree of change in note duration (*i.e.,* the standard deviation of the log of the event durations) in a melody. *Syncopation* estimates the degree of syncopation by assigning notes a strength in a metric hierarchy and averaging the strengths of all the notes in a melody; pulses are coded such that lower values are assigned to tones on metrically stronger beats. All three quantities increase the difficulty of perceiving or producing melodies (Eerola and North, 2000).

The mean success ratings for each stimulus were regressed on the predictor variables in a multiple regression analysis. The following pairwise correlations between the predictors were significant at $p < 0.05$: interval size, positively with interval dissonance ($r = 0.6$) and chromaticism ($r = 0.39$); harmonic closure, positively with chromaticism ($r = 0.49$); rhythmic variation, positively with syncopation ($r = 0.61$) and phrase length ($r = 0.73$); and rhythmic density, positively with syncopation ($r = 0.62$) and negatively with phrase length ($r = -0.54$). Because of this collinearity, in each analysis, redundant predictors were removed through backwards stepwise elimination using the Akaike Information Criterion: $AIC = n \log(RSS/n) + 2p + c$, for a regression model with p predictors and n observations, where c is a constant and RSS is the residual sum of squares of the model (Venables and Ripley, 2002). Since larger models

Predictor	β	Std. Error	t	p
Pitch Range	-0.29	0.08	-3.57	< 0.01
Pitch Centre	-0.21	0.10	-2.01	< 0.1
Interval Dissonance	-0.70	0.28	-2.54	< 0.05
Chromaticism	-0.27	0.03	-8.09	< 0.01
Phrase Length	-0.53	0.28	-1.91	< 0.1

Overall model: $R = 0.92$, $R^2_{adj} = 0.81$, $F(5, 22) = 25.04, p < 0.01$

Table 4: Multiple regression results for the mean success ratings of each test melody.

Stage	Feature Added	H
1	`Interval⊗Duration`	2.214
2	`ScaleDegree⊗Mode`	2.006
3	`ScaleDegree ⊗Int1stInPiece`	1.961
4	`Pitch⊗Duration`	1.943
5	`Thread1stInPhrase`	1.933
6	`ScaleDegree ⊗LastInPhrase`	1.925
7	`Interval⊗DurRatio`	1.919
8	`Interval⊗InScale`	1.917
9	`ScaleDegree⊗Duration`	1.912
10	`Int1stInPhrase`	1.911

Table 5: Results of feature selection for reduced information content over the dataset using an extended feature set.

provide better fits, this criterion balances model size, represented by p, with the fit of the model to the dependent variable, RSS.

More positive values of the predictors indicate greater deviation from the standards of the dataset (for pitch range and centre) or increased melodic complexity (for the remaining predictors), so we expect each predictor to show a negative relationship with the success ratings. The results of the multiple regression analysis with the mean success ratings as the dependent variable are shown in Table 4. The overall model accounts for approximately 85% of the variance in the mean success ratings. Apart from rhythmic structure, at least one predictor from each category made at least a marginally significant contribution to the fit of the model. Coefficients of all the selected predictors are negative as predicted. Overall, the model indicates that the judged success of a stimulus decreases as its pitch range and centre depart from the mean range and centre of the dataset, with increasing numbers of dissonant intervals and chromatic tones and if it has unequal phrase lengths.

4.2 Improving the Computational Systems

The constraints identified above mainly concern pitch range, intervallic structure and tonal structure. It seems likely that the confusion of relative minor and major modes is due to the failure of any of the Systems to represent mode. To examine this hypothesis, a linked feature `ScaleDegree⊗Mode` was added to the feature space. Furthermore, we hypothesise that the skewed distribution of pitch classes at phrase beginnings and endings can be better modelled by two linked features `ScaleDegree⊗1stInPhrase` and `ScaleDegree⊗LastInPhrase`. On the hypothesis that intervallic structure is constrained by tonal structure, we included another linked feature `Interval⊗InScale`.

System D: *Jesu, meiner Seelen Wonne*

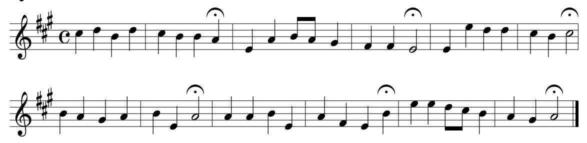


Figure 2: Melody generated by System D, based on the same chorale as Figure 1.

To examine whether the Systems can be improved to respect such constraints, we added the four selected features to the feature selection set used for System C. We ran the same feature selection algorithm over this extended feature space to select feature subsets which improve prediction performance; the results are shown in Table 5. In general, the resulting multiple-feature System, D, shows a great deal of overlap with System C. Just three of the nine features present in System C were not selected for inclusion in System D: `ScaleDegree⊗1stInBar`; `ThreadTactus`; and `Int1stInPiece`. This is probably because three of the four new features selected for inclusion in System D, were strongly related: `ScaleDegree⊗Mode`; `ScaleDegree⊗LastInPhrase`; and `Interval⊗InScale`. The first two of these, in particular, were selected early in the selection process; the existing feature `Int1stInPhrase` was added in the final stage. Ultimately, System D exhibits a lower average information content ($H = 1.91$) than System C ($H = 1.95$) in predicting unseen compositions in the dataset. The significance of this difference was confirmed by paired t tests over all 185 chorale melodies: $t(184) = 6.00, p < 0.01$, and averaged for each 10-fold partition of the dataset: $t(9) = 12.00, p < 0.01$.

4.3 A Melody Generated by System D

We now present preliminary results on System D's capacity to generate stylistically successful chorale melodies. System D uses the features in Table 5; it exhibits significantly lower entropy than System C in predicting unseen melodies. We used it to generate several melodies, as described above, with the same base melodies.

Figure 2 shows System D's most successful melody, based on Chorale 365. Its tonal and melodic structure are much more coherent than System C's melodies. Our multiple regression model, developed above to account for the judges' ratings of stylistic success, predicts that this melody would receive a rating of 6.4 on a seven-point scale of success as a chorale melody. While this result is positive, other melodies were less successful; System D must be analysed using our method to examine its ability to *consistently* compose stylistically successful melodies.

5 Discussion and Conclusions

Our statistical finite context grammars did not meet the computational demands of chorale melody composition, regardless of the representational primitives used. Since we attempted to address the limitations of previous context-modelling approaches to generating music, we might conclude that more powerful grammars are needed for this task. However, other approaches are possible. Further analysis of the capacities of finite context modelling systems may prove fruitful: future research should use the methodology developed here to analyse System D, and identify and correct its weaknesses. The MCMC generation algorithm may be responsible for failure, rather than the limitation of the models to finite context representations of melodic structure: more structured generation strategies, such as pattern-based sampling techniques, may be able to conserve phrase-level regularity and repetition in ways that our Systems were not.

Our evaluation method also warrants discussion. The adapted CAT yielded insightful results for ratings of stylistic success even though the judges were encouraged to rate the stimuli according to an absolute standard (*cf.* Amabile, 1996). However, the results suggest possible improvements: first, avoid any possibility of method artefacts by randomising the presentation order of both test and practice items for each judge and also the order in which rating scales are presented; second, the judges' comments sometimes reflected aesthetic judgements, so they should also give ratings of aesthetic appeal, to delineate subjective dimensions of the product domain in the assessment (Amabile, 1996); and third, though influence of prior familiarity with the test items was ambiguous, bias resulting from recognition should be avoided.

Our results suggest that the task of composing a stylistically successful chorale melody presents significant challenges as a first step in modelling cognitive processes in composition. Nonetheless, our evaluation method proved fruitful in examining the generated melodies in the context of existing pieces in the style. It facilitated empirical examination of specific hypotheses about the models through detailed comparison of the generated and original melodies on several dimensions. It also permitted examination of objective features of the melodies which influenced the ratings and subsequent identification of weaknesses in the Systems and directions for improving them. This practically demonstrates the utility of analysis by synthesis for evaluating cognitive models of composition—if it is combined with an empirical methodology for evaluation such as that developed here.

References

Amabile, T. M. (1996). *Creativity in Context*. Westview Press, Boulder, Colorado.

Ames, C. and Domino, M. (1992). Cybernetic Composer: An overview. In Balaban, M., Ebcioğlu, K., and Laske, O., editors, *Understanding Music with AI: Perspectives on Music Cognition*, pages 186–205. MIT Press, Cambridge, MA.

Baroni, M. (1999). Musical grammar and the cognitive processes of composition. *Musicæ Scientiæ*, 3(1):3–19.

Cleary, J. G. and Teahan, W. J. (1997). Unbounded length contexts for PPM. *The Computer Journal*, 40(2/3):67–75.

Conklin, D. (2003). Music generation from statistical models. In *Proceedings of the AISB 2003 Symposium on Artificial Intelligence and Creativity in the Arts and Sciences*, pages 30–35, Brighton, UK. SSAISB.

Conklin, D. and Witten, I. H. (1995). Multiple viewpoint systems for music prediction. *Journal of New Music Research*, 24(1):51–73.

Eerola, T. and North, A. C. (2000). Expectancy-based model of melodic complexity. In Woods, C., Luck, G., Brochard, R., Seddon, F., and Sloboda, J. A., editors, *Proceedings of the Sixth International Conference on Music Perception and Cognition*, Keele, UK. Keele University.

Johnson-Laird, P. N. (1991). Jazz improvisation: A theory at the computational level. In Howell, P., West, R., and Cross, I., editors, *Representing Musical Structure*, pages 291–325. Academic Press, London.

Krumhansl, C. L. (1990). *Cognitive Foundations of Musical Pitch*. Oxford University Press, Oxford.

MacKay, D. J. C. (1998). Introduction to Monte Carlo methods. In Jordan, M. I., editor, *Learning in Graphical Models*, NATO Science Series, pages 175–204. Kluwer Academic Press, Dordrecht, The Netherlands.

Manzara, L. C., Witten, I. H., and James, M. (1992). On the entropy of music: An experiment with Bach chorale melodies. *Leonardo*, 2(1):81–88.

Marsden, A. (2000). Music, intelligence and artificiality. In Miranda, E. R., editor, *Readings in Music and Artificial Intelligence*, pages 15–28. Harwood Academic Publishers, Amsterdam.

Pearce, M. T. (2005). *The Construction and Evaluation of Statistical Models of Melodic Structure in Music Perception and Composition*. PhD thesis, Department of Computing, City University, London, UK.

Pearce, M. T., Meredith, D., and Wiggins, G. A. (2002). Motivations and methodologies for automation of the compositional process. *Musicae Scientiae*, 6(2):119–147.

Pearce, M. T. and Wiggins, G. A. (2006). Expectation in melody: The influence of context and learning. *Music Perception*, 23(5):377–406.

Ponsford, D., Wiggins, G. A., and Mellish, C. (1999). Statistical learning of harmonic movement. *Journal of New Music Research*, 28(2):150–177.

Rabiner, L. R. (1989). A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285.

Temperley, D. (1999). What's key for key? The Krumhansl-Schmuckler key-finding algorithm reconsidered. *Music Perception*, 17(1):65–100.

Venables, W. N. and Ripley, B. D. (2002). *Modern Applied Statistics with S*. Springer, New York.

von Hippel, P. T. (2000). Redefining pitch proximity: Tessitura and mobility as constraints on melodic intervals. *Music Perception*, 17(3):315–127.

SYSTEMATIC EVALUATION AND IMPROVEMENT OF STATISTICAL MODELS OF HARMONY

Raymond P. Whorley **Geraint A. Wiggins** **Marcus T. Pearce**
Centre for Cognition, Computation and Culture
Goldsmiths, University of London
New Cross, London SE14 6NW, U.K.
{r.whorley,g.wiggins,m.pearce}@gold.ac.uk

Abstract

We are investigating the utility of Markov models in relation to the learning of the task of four-part harmonisation, which is a creative musical activity. A program is described which uses statistical machine learning techniques to learn this task from a suitable corpus of homophonic music. The task is decomposed into a series of more manageable sub-tasks; these are each modelled by Markov models, which can use contexts drawn from symbols describing past, current and future chords. The results of a number of initial studies, for example comparing different types of model and the effect of corpus size, are given. There is also some discussion about harmonisations that have been generated by the program by random sampling of the probability distributions in the models. Following this, a procedure for the systematic evaluation and "optimisation" of the sub-task models, involving the application of an information-theoretic measure, is presented, along with some more results. An appraisal of the procedure's shortcomings is made, and ideas for its improvement are put forward. Finally, an indication of the future direction of the work (which is currently in its early stages) is given.

Keywords: Machine learning, harmonisation, Markov models, evaluation.

1 Introduction

The current work contributes to the study of computational creativity in music by seeking to improve the computational simulation of four-part harmonisation, which is a creative human activity. We are investigating the utility of Markov models in relation to the learning of this task, in pursuit of which a program is being developed which seeks to learn a style of four-part harmonisation from a suitable corpus of homophonic music by means of statistical machine learning techniques. There are two primary motivations for this work. The first is that the models so produced should be of use in addressing stylistic and musicological issues, and the second is that the models could form the basis of a cognitive model of harmonic perception (Pearce et al., 2002). It is intended that the program will eventually handle music which is not completely homophonic, such as the set of chorale melodies harmonised by J. S. Bach.

The approach adopted breaks the overall harmonisation task into a series of more manageable sub-tasks (Hild et al., 1992; Allan, 2002) . The sub-tasks implemented so far relate to the harmonic function of the chords; sub-tasks that for example assign particular notes to the alto, tenor and bass parts will follow in due course. An information-theoretic measure is used to evaluate the various sub-task models (Allan and Williams, 2005), and a systematic approach to evaluation is taken which guides the creation of the models (Pearce, 2005). Having learned a style of harmonisation in this way, the program is equipped to generate harmonisations to "unseen" melodies.

It should be noted that this is very much "work in progress"; indeed, work at a very early stage. The results obtained so far can only therefore be taken as indicative, rather than definitive.

2 Corpus, Test Data and Input

The training corpus currently comprises fifteen major key hymn tunes with harmonisations taken from Vaughan Williams (1933) and Nicholson et al. (1950). In addition, there are two sets of five major key test melodies with harmonisations from the same sources (although only three of these melodies are used for initial studies).

All of the training corpus melodies, along with a certain amount of annotation (Ponsford et al., 1999), are assigned to a single sequence (array) of symbols. Annotation symbols are placed at the beginning and end of each melody, and at phrase boundaries. All of the training corpus bass parts are assigned to another symbol sequence in a completely analogous way. Melody and bass notes are normalised to seven *scale degree* symbols which are not dependent upon key; for example a tonic note is represented by "1" and a dominant note by "5". These scale degrees can be chromatically altered by post-fixing with "#" or "b" (sharp or flat). No account is currently taken of note duration or metrical importance.

Symbols representing the harmonic function of chords in the training corpus (determined by the principal author) are assigned, along with annotation symbols, to yet another sequence. The chords are normalised to root position triads and seventh chords; for example a tonic chord is represented by "I" and a supertonic seventh chord by "ii7". Chords containing chromatically altered notes are deemed to be triads or seventh chords in the nearest possible related key; for example if a chord containing the notes D, $F\sharp$ and A appears when the key is C major, it is represented by "V/V" (dominant of the dominant). Since this program does not keep track of modulations, the chord could, alternatively, be described as "II", a major triad on the second degree of the original key. The latter approach requires less music-theoretic knowledge and gives an equally good description; therefore it might be adopted later. On the other hand, in their multiple viewpoint technique, Conklin and Cleary (1988) propose a viewpoint to detect modulations; something of this sort could be incorporated into the program. Note that since the hymns are homophonic (*i.e.*, consisting of block chords only, with no extra-chord movement such as passing notes), it is simple to associate all symbols comprising a particular chord with the same array index for ease of retrieval.

Test melodies, bass parts and harmonic symbols are assigned to sequences in the same way as for the training corpus, except that they are individually assigned. Only one test melody/harmony combination is used per program run.

A second version of the harmonic symbol sequence also exists, which contains chord inversion information in the symbols. This is used for purposes of comparison.

3 The Models and Their Construction

3.1 Markov Models

Markov chains, or *n-gram models*, widely used in natural language processing (Manning and Schütze, 1999), are sequences of n symbols (*e.g.*, a sequence of musical note names), where it is assumed that the probability of the last symbol appearing is dependent upon the previous $n-1$ symbols (the *context*). This is an approximation of the probability of the last symbol being dependent upon its entire history; it can therefore be expected that the longer the context, the more accurate the *transition probabilities*. The size of the context is known as the *order* of the Markov chain; therefore for example a 2-gram (or *bigram*) is first-order (*n.b.*, a Markov "chain" with no context, a *unigram*, is zeroth-order).

Markov chains have been used in the prediction of both harmony (Ponsford et al., 1999) and melody (Pearce, 2005). Allan (2002) tries using Markov models with additional context to predict harmonic symbols, but decides that it would be better to use hidden Markov models. The contexts he uses, however, are restricted to past and current melodic symbols, and past harmonic symbols. Bearing in mind that a composer can look ahead while harmonising a melody, we extend the scope of possible contexts to include future symbols (*i.e.*, those to the right of the melody note currently being harmonised) wherever they exist at the various stages of the harmonisation process; for example, future melodic symbols can be used from the beginning of the process.

3.2 Smoothing

It is often the case that contexts occurring while, for example, generating a harmony to an "unseen" melody, cannot be found in the model. In order to be able to make informed predictions in these circumstances, a number of models using different contexts are used in conjunction with a *smoothing* technique. Conklin and Cleary (1988) and Allan (2002) use a simple method, which has worked well with text, known as *back-off* smoothing (Katz, 1987). For Markov chains, a number of models of different order are created. When searching for a particular context, the highest order model is checked first. If the context is not found, the second-highest order model is checked, and so on, until the (progressively smaller) context is found. For the "mixed" contexts that we are using, however, back-off is not limited to successive contexts of decreasing size; different contexts of the same overall size can be involved in the back-off sequence.

It should be noted that an *escape* method is normally associated with back-off smoothing, whereby some of the probability mass is assigned to unseen Markov chains, the probabilities of which are estimated during back-off. No such escape method has so far been implemented here; if a context is not found in a particular model, then the model is discarded. The next model to be tried is assumed to contain all of the probability mass.

There are other more complicated smoothing methods that are also known to work well, but one of the purposes of this paper is to investigate how an optimum, or close to optimum, back-off sequence can be determined, and how well it performs.

3.3 Decomposition of the Harmonisation Task

Allan (2002) follows Hild et al. (1992) in breaking up the harmonisation task into three sub-tasks, in the following order: "harmonic skeleton", "chord skeleton" and "ornamentation". In the "harmonic skeleton" sub-task, harmonic symbols are assigned to each beat; the actual notes of each chord are filled out in the "chord skeleton" sub-task; and additional notes such as passing notes are supplied during "ornamentation". Wiggins (1998) suggests that the problem should be broken up even further, by for example choosing cadences first.

Here, we break up the task of assigning harmonic symbols to melody notes as much as possible. The training corpus has been augmented with symbols representing root position triads and seventh chords (see Section 2). The program assigns these root position harmonic symbols on a first pass, and then bass note scale degree symbols during a second pass, thereby effectively assigning chord and inversion information. This is preferred to the use of harmonic symbols incorporating inversion information, in keeping with the dictum of using as little music-theoretic knowledge as possible in the training of such models. Knowledge about inversions is effectively induced from the training corpus as the models learn the sequential structure of the given symbolic representations. Having said that, the corpus has also been augmented with harmonic symbols which incorporate chord inversion information (see Section 2), for purposes of comparison. It should be noted that in this case the appropriate bass note scale degrees still need to be induced from the corpus, since no logic has been encoded to interpret the harmonic symbols; however, they can be determined with a probability of 1.0 during the second pass for all harmonic symbols that have been "seen" in the corpus.

Each of the two passes is broken up into three subtasks, each requiring its own model. The first deals with cadence chords, which have been taken to be the last two chords in each phrase; the second is concerned with all chords except the final three in each phrase; and the third specialises in the antepenultimate chord of each phrase, with the objective of knitting together the bulk of the harmony in each phrase with its cadence.

3.4 Model Construction

Cadences are important structural components of music, and as such deserve particular attention. The idea here is to create models of symbols used at cadences, with the intention not only of generating acceptable cadences in isolation, but also generating appropriate sequences of cadences (*e.g.*, avoiding too many cadences of the same type following each other). The sub-task which predicts root position harmonic symbols for melody notes in cadential positions will be used to exemplify the construction of models.

The first step is to create a sequence from pairs of melodic scale degree symbols at the end of each phrase, along with all of the annotation symbols. Harmonic symbols are treated in a completely analogous way; thus sequences containing only symbols at cadential positions (plus annotations) have been formed.

Markov models are then constructed using *maximum likelihood estimation* (Manning and Schütze, 1999), recording the probability of a particular harmonic symbol appearing given a particular context. The context is taken from the cadential symbol sequences created above. A series of models is created, each successive one having either a different context of the same size or a smaller context compared with the one before, going all the way down to no context at all (zeroth-order overall).

Models for the other five sub-tasks are constructed in a similar way. The main difference is that for the four models not directly concerned with cadences, all of the input symbols are used (even for the pre-cadence model, in spite of being used for only one chord per phrase).

To begin with, the back-off sequences were fairly arbitrary, consisting mainly of a succession of ever-smaller contexts. Some initial studies were performed using these back-off strategies, before using an information-theoretic measure to guide the construction of back-off sequences that are likely to be much closer to optimal.

4 Evaluation Study

4.1 Evaluation Method

Some initial studies were performed using arbitrary back-off strategies. The same information-theoretic measure used later to improve these strategies, *cross-entropy*, was used here both to evaluate models, using test data, and to compare generated harmonies. If we define $P_m(S_i|C_{i,m})$ as the probability of the i^{th} musical symbol given its context for a particular model m, and assume that there are a total of n sequential symbols, then cross-entropy is given by $-(1/n)\sum_{i=1}^{n}\log_2 P_m(S_i|C_{i,m})$. It is a useful measure, because it gives a "per symbol" value; it can therefore be used to compare sequences of any length. All else being equal, the model which assigns the lowest cross-entropy to a test sequence is the best descriptor of the data (Allan, 2002). This is equivalent to saying that the model which assigns the highest geometric mean of the conditional probabilities in a test sequence is the best descriptor of the data.

4.2 Description of Studies

The studies compared two broadly different model types; one using root position harmonic symbols, and the other employing harmonic symbols incorporating inversion information. (Note that in each case, two passes of a melody are required; one to predict harmonic symbols and the other to predict bass note scale degree symbols. See Section 3.3). Within each of these models, the effect of corpus size on both test data and generated harmony was investigated. Three hymns were used as test data: St. Anne, Winchester Old and Winchester New. The melodies of these same three hymns were also used in the random generation of harmony (*i.e.*, these melodies were automatically harmonised by the program, sampling from the statistical models created from the training corpus). Each melody was harmonised ten times, and an overall cross-entropy for the three melodies was calculated for each pass (*i.e.*, for allocation of harmonic symbols and bass note scale degree symbols). The results are summarised in Table 1.

4.3 Results

In order to achieve a degree of brevity in the following discussion, the model using root position harmonic symbols will be referred to as the *root position* model, and the one using harmonic symbols incorporating inversion information will be referred to as the *inversions* model. It can be seen from Table 1 that the root position model better describes the test data (*i.e.*, it has lower overall cross-entropies), in spite of the inversions model having a cross-entropy of zero for the second pass. It should be borne in mind, however, that the arbitrary back-off strategies might have favoured one model over the other; therefore a new comparison should be made after optimising (as far as is practical) the back-off sequences for each model. A start will be made on this in the next section.

It is to be expected that models benefit from larger training corpora, since more previously "unseen" contexts

Table 1: Cross-entropy comparisons across model type, corpus size and test data/generated harmony

Model type	Corpus size	Test data			Generated harmony		
		Pass 1	Pass 2	Passes 1 & 2	Pass 1	Pass 2	Passes 1 & 2
Inversions	5	1.448	0.000	0.724	0.508	0.000	0.254
	10	1.132	0.000	0.566	0.288	0.000	0.144
	15	1.096	0.000	0.548	0.296	0.000	0.148
Root position	5	0.894	0.475	0.685	0.378	0.130	0.254
	10	0.787	0.276	0.531	0.220	0.119	0.169
	15	0.817	0.240	0.529	0.240	0.078	0.159

are likely to come to light with each addition to the corpus. The figures in the table bear this out; but it should be noted that although in general there is an improvement between corpus sizes of ten and fifteen hymns, it is very small compared with the marked improvement between five and ten hymns. One possible reason for this is that most of the "knowledge" concerning the basic harmonisation of homophonic hymns can be garnered from a relatively small corpus. Another possibility is that although there might be additional information in larger corpora, the system is, for one reason or another, unable to model it. A third possibility is that the additional five hymns introduce new structure that is mostly not found in the test data, resulting in only a very small improvement. The latter possibility can easily be tested by further increasing the corpus size, using a larger or different set of test data, or all of the above. One way of testing the other two possibilities is to repeat the studies after completion of the improved back-off strategies.

There is a large disparity between the cross-entropy figures for test data and generated harmony; the figures for generated harmony are much lower. We shall look at this issue in more detail in Section 5.

4.4 Discussion of Generated Harmony

Having harmonised a melody, the program outputs triples of melody note scale degree, harmonic symbol and bass note scale degree. In order to listen to realisations of these basic outputs, MIDI files are created "by hand". Wherever a chord chosen by the program is in agreement with the hymnal harmony, the same arrangement of notes within the chord as appears in the hymnal is used in the MIDI file, provided that the usual voice-leading rules can be satisfied.

The harmonies are often quite good, especially the ones with lower cross-entropies. As an indication of this (using the root position model) the harmonies with the lowest cross-entropy for Winchester Old (five- and ten-hymn corpora) and Winchester New (fifteen-hymn corpus: see Figure 1) contain between 54 and 57 percent of chords that are exactly the same as in the hymnal. Other chords are good substitutions (*e.g.*, the antepenultimate chord in Figure 1, a root position subdominant, is a perfectly good alternative to the first inversion supertonic seventh in the hymnal), while yet others sound distinctly out of place (*e.g.*, the second inversion chord at the end of the first full bar of Figure 1). In contrast, the higher cross-entropy harmonies (see Figure 2) sound further removed from the style of the corpus, often containing chord progressions that are displeasing to the ear (*e.g.*, the first two chords of the second full bar of Figure 2 inescapably contain parallel octaves, and the progression from the last chord of the third full bar of Figure 2 to the following chord sounds far from smooth).

4.5 Ten-fold Cross-validation

In addition to the above studies, a *ten-fold cross-validation* was performed for both the root position and inversions models. Each of the hymns in turn was removed from the ten-hymn corpus and used as test data for a model trained using the remaining nine hymns. The results are summarised in Table 2. The most obvious difference between the two models is the number of harmonic symbols encountered in the test data that had not been observed in the training corpus; there are far fewer such symbols associated with the root position model. This is unsurprising, bearing in mind that there are far fewer root position harmonic symbols to discover.

Overall, the figures confirm the earlier finding that the root position model gives a better description of the test data (of course, the same proviso about the arbitrary back-off strategies applies). In this case, it is due at least in part to the greater number of unseen harmonic symbols, which are assigned a very low probability. In addition, when an unseen harmonic symbol is encountered, the model has little or no idea which bass note scale degree to assign to that chord, resulting in another low probability. The final thing to note from this table is that overall, there is a narrower spread of cross-entropies for the root position model. For example, the difference between the highest and lowest cross-entropies for the root position model is 0.823, compared with 1.018 for the inversions model; this superior consistency might also be an indication that the root position model is better.

5 Systematic Improvement of Back-off Strategies

It should be noted from the beginning that the method described below is not guaranteed to produce an optimal back-off sequence; the method is less than rigorous in several respects. It is, however, expected to result in a close to optimal solution.

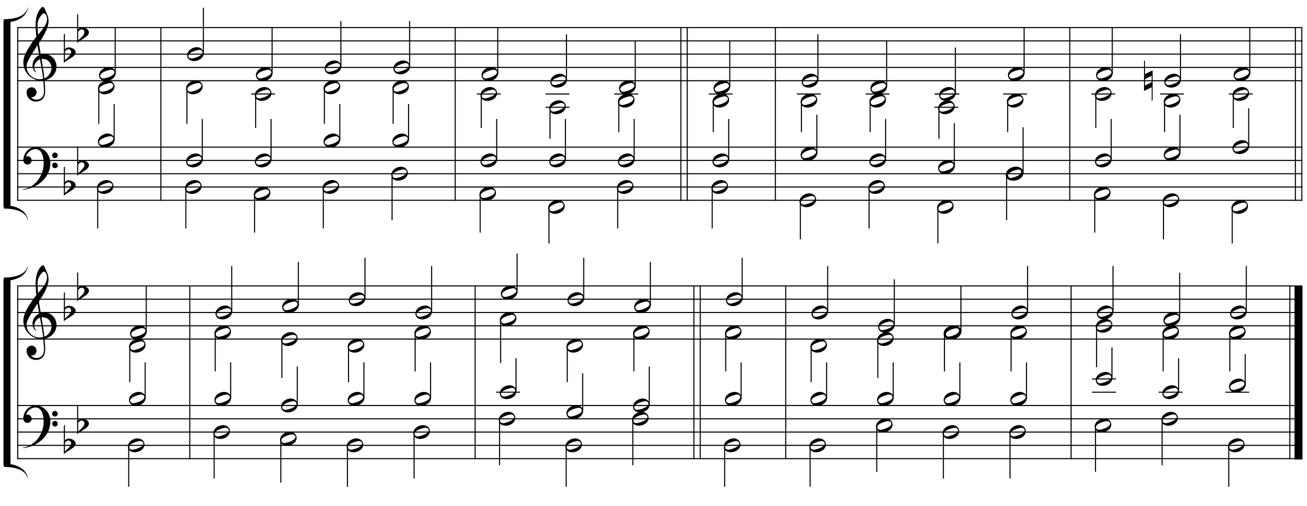


Figure 1: Low cross-entropy harmonisation of Winchester New, using the root position model trained on a fifteen-hymn corpus, with an arbitrary back-off strategy

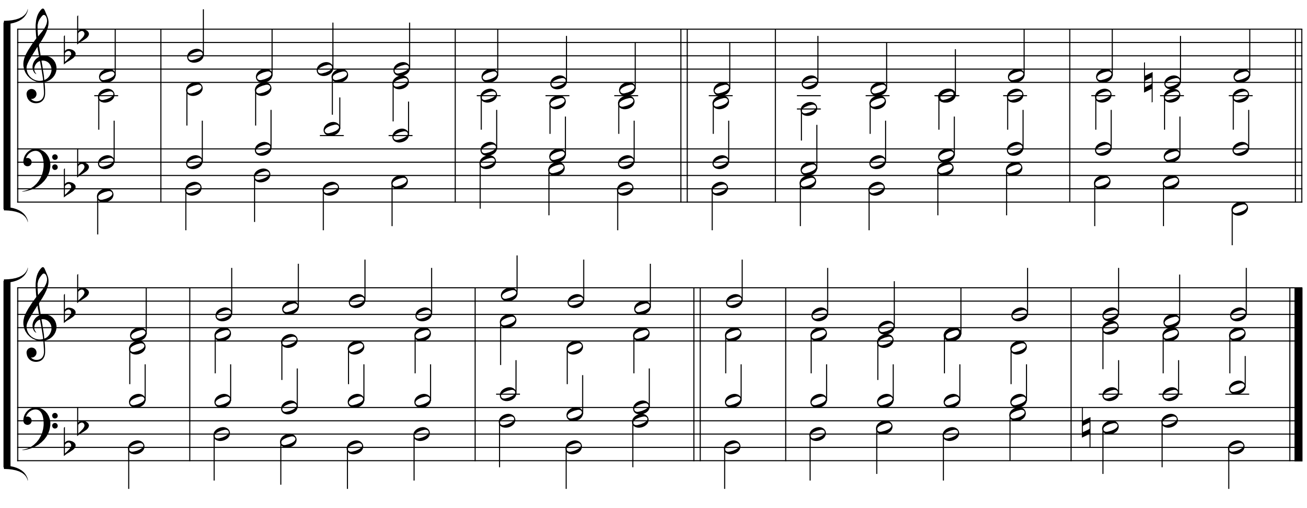


Figure 2: High cross-entropy harmonisation of Winchester New, using the root position model trained on a fifteen-hymn corpus, with an arbitrary back-off strategy

Table 2: Cross-entropies from ten-fold cross-validation: inversions and root position models

	Inversions model				Root position model			
Hymn	Pass 1	Pass 2	Passes 1 & 2	"Unseen" symbols	Pass 1	Pass 2	Passes 1 & 2	"Unseen" symbols
1	2.708	0.326	1.517	3	1.413	0.908	1.160	1
2	1.141	0.099	0.620	1	0.674	0.468	0.571	1
3	2.610	0.515	1.563	9	1.697	0.823	1.260	3
4	2.260	0.372	1.316	4	1.800	0.685	1.242	2
5	1.753	0.099	0.926	1	0.893	0.480	0.686	0
6	2.281	0.127	1.204	3	1.336	0.895	1.115	2
7	1.740	0.080	0.910	1	0.932	0.363	0.648	0
8	1.460	0.078	0.769	1	0.895	0.477	0.686	1
9	1.089	0.000	0.545	0	0.637	0.236	0.437	0
10	1.861	0.302	1.081	1	1.175	0.810	0.993	0
Overall	1.977	0.227	1.102	24	1.219	0.655	0.937	10

5.1 Procedure Description

First of all, two new hymns are added to the test data, making five in all. Having more test data will tend to smooth out any statistical anomolies. In the first instance, the back-off strategies to be improved are those of the root position model, using a fifteen-hymn corpus. In order to set a baseline standard, cross-entropies for the two passes are initially calculated using zeroth-order models for all of the sub-tasks. Following this, for each of the sub-task models in turn, and with all of the other sub-task models set to zeroth-order, all relevant first-order contexts are evaluated (in each case backing off to zeroth-order), and then a back-off sequence is assembled for a complete first-order sub-task model. The sequence starts with the context achieving the lowest cross-entropy and ends with no context at all, which has the highest cross-entropy.

For example, the single-symbol contexts relevant to the sub-task model assigning harmonic symbols to the majority of melody notes in each phrase are the previous harmonic, previous melodic, current melodic and future melodic symbols. It turns out that the current melodic symbol achieves by far the lowest cross-entropy (which is no great surprise), followed by the past harmonic, past melodic and future melodic symbols. The back-off sequence is therefore assembled in this order, finishing off with no context. Generally speaking, there will be no back-off from the current melodic context; but on rare occasions a previously "unseen" melodic symbol might be encountered, necessitating back-off (*e.g.*, the melodic symbol "2b" does not exist in the current corpus, but could theoretically occur).

5.2 Assumptions and Restrictions

Once all of the sub-task models have been through this process, work can begin on marshalling the second-order contexts. In order to avoid searching a huge number of contexts (the number of possible different contexts increases dramatically with overall context size) we have assumed that the best combinations (*i.e.*, those that achieve the lowest cross-entropies) will contain the best context from the next-lowest order of model. In the case of our example sub-task model, therefore, the relevant dual-symbol contexts all contain the current melodic symbol in addition to the past harmonic, past melodic and future melodic symbols respectively. Once again, all of the other sub-task models are set to zeroth-order, and the sub-task model under consideration backs off directly from a dual-symbol context to zeroth-order. We have assumed that a larger context which subsumes a smaller one will always be better than that smaller context, even if its cross-entropy is higher, so that the new contexts need only be added to the head of the back-off chain. There is some logic to this. It is likely that smaller contexts will be matched more often than larger ones; therefore even though the resulting individual probabilities might be lower, they can easily contribute to a greater improvement in the overall cross-entropy by virtue of the number of symbols assigned. The assignment of fewer, but higher-probability symbols earlier in the back-off sequence will improve things further.

The process continues for an arbitrary number of overall context sizes. It should be pointed out at this stage that another simplifying restriction that we have imposed is that symbols can only be added to the context such that constituent parts of the context form chains, radiating outwards from the chord which is the focus of attention at the time. For example, if the context contains harmonic symbol $i - 1$ (immediately preceding current chord i), and another past harmonic symbol is to be added, it must be harmonic symbol $i - 2$ in the sequence from which the context is taken.

The above assumptions, and issues around them, mean that the method is not guaranteed to produce an optimum solution. It is conceivable that larger contexts not containing the best context from the previous stage of the process could achieve a lower cross-entropy; this can be tested. Even if this were not the case, however, the poorer contexts could still contribute by being inserted somewhere further down the back-off sequence. It is also conceivable that some larger contexts that do contain the best previous context might actually contribute to a deterioration in performance. This can be guarded against by testing contexts in conjunction with the full previously assembled back-off sequence for a sub-task model.

Probably the best way of improving this "optimisation" method, however, is to calculate cross-entropies only for the assignments that a particular context makes. It will then be perfectly clear, for example, that a model assigning only one harmonic symbol to a melody note with a probability of 0.9 should be higher up the back-off sequence than a model assigning harmonic symbols to ten melody notes, each with a probability of 0.8.

5.3 Results

The results are summarised in Table 3. Evaluation is carried out using a set of test data comprising harmonisations, found in hymnals, of five different hymn tunes. These hymn tunes do not appear in the training corpus or in the test data used in the "optimisation" process; they are completely "unseen" from the model's point of view. Cross-entropies for the test data used during the "optimisation" procedure are also shown for comparison. The generated harmony results are derived from harmonising the set of melodies in the evaluation test data, since these melodies are still "unseen" on any given run of the program.

As expected, there is a huge improvement in the model between zeroth-order and first-order. There is a much smaller, but still very significant improvement between first-order and second-order. Unfortunately, due to the fact that results have only so far been produced for three context sizes, it is difficult to judge how the trend in improvement will continue. The cross-entropy of 0.880 ("unseen" test data) for the second order model is significantly higher than the 0.529 for the model with arbitrary back-off. This latter model, however, has much longer back-off sequences in terms of overall context size (although it has far fewer instances of different contexts of the same size).

Another thing to notice is that the models predicting bass note scale degrees (pass 2) perform significantly better than those predicting harmonic symbols (pass 1). This is also true for the model with arbitrary back-off. Finally,

Table 3: Cross-entropy comparisons across context size and test data/generated harmony for a close to optimal root position model trained on a fifteen-hymn corpus

Model type	Context size	"Optimisation" test data			"Unseen" test data			Generated harmony		
		Pass 1	Pass 2	Passes 1 & 2	Pass 1	Pass 2	Passes 1 & 2	Pass 1	Pass 2	Passes 1 & 2
Root position	0	2.743	2.573	2.658	2.532	2.458	2.495	2.896	2.542	2.719
	1	1.399	0.691	1.045	1.218	0.907	1.062	1.538	0.812	1.175
	2	1.070	0.478	0.774	1.048	0.712	0.880	0.960	0.596	0.778

the cross-entropies of the generated harmony are similar to those of the test data. This is in stark contrast with the results of the model with arbitrary back-off, which shows much lower cross-entropies for the generated harmony. It seems likely that the similarity in this case, and dissimilarity in the former case, directly result from the very different back-off strategies employed.

6 Conclusions and Future Work

We have discussed ways in which Markov models used for learning the creative human task of four-part harmonisation may be improved. Some preliminary studies have been carried out, and the results presented.

The root position model describes the test data better than the inversions model when the arbitrary back-off strategies are employed; since it is possible that these strategies are biased towards the root position model, another comparison will be made using back-off strategies thought to be close to optimal for each of the models.

Models using the arbitrary back-off strategies improve significantly between corpus sizes of five and ten hymns, but only very slightly between ten and fifteen hymns. The corpus size will be increased further to test whether this is a statistical aberration or a real effect. The studies will be repeated using close to optimal back-off strategies to find out if more "knowledge" can be extracted. This time, for the generation side of the studies, hymn tunes not in the test data (nor, of course, in the training corpus) will be used, since the test data will have been employed for "optimisation" purposes.

Models using close to optimal back-off strategies improve greatly between zeroth-order and first-order. There is a much smaller, but still very significant improvement between first-order and second-order. Further work will be done on the improvement of back-off strategies, including: extending to higher-order contexts; calculating cross-entropies only for the assignments that a particular context makes; and investigating the effectiveness of contexts not containing the previous best context.

Future work also includes comparing the performance of different arrangements of the root position model; for example reversing the passes such that bass note scale degree symbols are assigned before harmonic symbols, and assigning both symbols on a chord by chord basis rather than in two complete passes of the melody. An investigation into the effect of increasing the number of chords covered by the cadential models is also proposed, as well as ascertaining whether or not the pre-cadential models really make a useful contribution.

Consideration of the effect of metre and note duration is also planned for the not-too-distant future. Clearly, extending the system to predict actual bass notes (rather than just the scale degree), inner parts and, for example, passing notes, is a little further in the future.

References

Allan, M. (2002). Harmonising chorales in the style of Johann Sebastian Bach. Master's thesis, School of Informatics, University of Edinburgh.

Allan, M. and Williams, C. K. I. (2005). Harmonising chorales by probabilistic inference. In L. K. Saul, Y. W. and Bottou, L., editors, *Advances in Neural Information Processing Systems*, volume 17. MIT Press.

Conklin, D. and Cleary, J. G. (1988). Modelling and generating music using multiple viewpoints. In *Proceedings of the First Workshop on AI and Music*, pages 125–137. The American Association for Artificial Intelligence.

Hild, H., Feulner, J., and Menzel, W. (1992). Harmonet: A neural net for harmonizing chorales in the style of J. S. Bach. In R. P. Lippmann, J. E. M. and Touretzky, D. S., editors, *Advances in Neural Information Processing Systems*, volume 4, pages 267–274. Morgan Kaufmann.

Katz, S. M. (1987). Estimation of probabilities from sparse data for the language model component of a speech recogniser. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35(3):400–401.

Manning, C. D. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. The MIT Press.

Nicholson, S., Knight, G. H., and Dykes Bower, J., editors (1950). *Hymns Ancient & Modern Revised*. William Clowes and Sons, Ltd.

Pearce, M. T. (2005). *The Construction and Evaluation of Statistical Models of Melodic Structure in Music Perception and Composition*. PhD thesis, Department of Computing, City University, London.

Pearce, M. T., Meredith, D., and Wiggins, G. A. (2002). Motivations and methodologies for automation of the compositional process. *Musicae Scientiae*, 6(2):119–147.

Ponsford, D., Wiggins, G. A., and Mellish, C. (1999). Statistical learning of harmonic movement. *Journal of New Music Research*, 28(2):150–177.

Vaughan Williams, R., editor (1933). *The English Hymnal*. Oxford University Press.

Wiggins, G. A. (1998). The use of constraint systems for musical composition. In *Proceedings of the 13th Biennial European Conference on Artificial Intelligence (ECAI) Workshop: Constraint techniques for artistic applications*, Brighton, UK.

Session 4
Applied Creative Systems

A practical application of computational humour

Graeme Ritchie
Computing Science
University of Aberdeen
Aberdeen AB24 3UE
gritchie@csd.abdn.ac.uk

Ruli Manurung[*] **Helen Pain**
School of Informatics
University of Edinburgh
Edinburgh EH9 8LW
ruli.manurung@ed.ac.uk
H.Pain@ed.ac.uk

Annalu Waller, Rolf Black, Dave O'Mara
School of Computing
University of Dundee
Dundee DD1 4HN
awaller@computing.dundee.ac.uk
rolfblack@computing.dundee.ac.uk
domara@computing.dundee.ac.uk

Abstract

The past 15 years has seen the development of a number of programs which perform tasks in the area of humour, but these have been exploratory research prototypes, usually on a very small scale, and none of them interacted with users. Amongst those which actually created humorous texts, the JAPE program was probably the most substantial, but even it was far from being useful for any practical purpose. We have developed a fully engineered riddle generator, inspired by the ideas in the JAPE system, which uses a large-scale multimedia lexicon and a set of symbolic rules to generate jokes. It has an interactive user interface, specially designed for children with complex communication needs (CCN), so that users can make choices to guide the riddle generator. The software is robust, stable, and responds sufficiently promptly that naive users can interact without difficulty. It has been tested over with real users (children with CCN), with highly positive results, and is publicly available for free download.

Keywords: Computational humour, riddles, AAC, joke generation

1 Introduction

Since 1992, research into computational humour has led to a number of exploratory implementations, including some joke-generation systems. However, these have generally been small exploratory research prototypes rather than full practical applications. We have developed a state of the art riddle-generation system which is completely usable by untrained and naive users, and which has been evaluated in a systematic manner. Our program (STANDUP - System To Augment Non-speakers Dialogue Using Puns) is aimed at young children, and lets them play with words and phrases by building punning riddles through a simple interactive user-interface.

A punning riddle is a question-answer joke in which the answer makes a play on words, as in (1).

(1) What kind of tree is nauseated?
A sick-amore.

What makes these jokes computationally manageable (and also makes them good illustrative examples of simple language mechanisms) is their reliance on simple linguistic relations such as homophony and synonymy.

Our target users are children with impaired speech and limited motor skills (as often results from cerebral palsy). Such *complex communication needs* (CCN) can result in lower levels of literacy than in typically-developing counterparts of the same age (Smith, 2005). This can prevent full involvement in normal forms of language play (Waller, 2006), leading to poorer skills in language, communication and social interaction (Lindsay and Dockrell, 2000). The STANDUP software is a "language playground", with which a child can explore sounds and meanings by making up jokes, with computer assistance. We conjecture that this will have a beneficial effect on literacy and communication skills, but our project addressed two more basic research questions:

(i) is it feasible to build an interactive riddle-generator which can be controlled by children with CCN?

(ii) if so, in what ways do such children use the software?

We have adopted the joke-construction mechanisms of the JAPE program (Binsted, 1996; Binsted and Ritchie, 1994, 1997) as the core of our stable, usable, robust, user-friendly, interactive system for children with CCN. In this paper, we will describe what was involved in developing a working application from the research ideas.

2 Computational Humour

At present, there is no theory of humour which is sufficiently precise, detailed and formal to be implementable. Hence computational humour has so far largely consisted of small research prototypes based on mechanisms designed specifically for whatever (narrow) problem was being tackled (for overviews, see Ritchie (2001b), Hulstijn and Nijholt (1996), Stock et al. (2002)). Many of these

[*] Now at Faculty of Computer Science, Universitas Indonesia, Depok 16424, Indonesia. maruli@cs.ui.ac.id

programs have been generators of simple verbal jokes, and have been very small studies (often student projects).

Lessard and Levison (1992) built a program which created a simple type of pun, the *Tom Swifty*. Lessard and Levison (1993) sketch the workings of a program which produced some basic forms of punning riddle. Venour (1999) built a small program which generated simple texts consisting of a one-sentence set-up and a punning punchline consisting of a head noun preceded by an adjective or a noun modifier. All these used an existing natural language generator, VINCI (Levison and Lessard, 1992). The WISCRAIC program (McKay, 2002) produced simple puns in three different linguistic forms (question-answer, single sentence, two-sentence sequence).

These systems operated with small amounts of hand-crafted data, and were not given much serious testing. The Lessard and Levison projects report no performance or evaluation, while Venour and Mackay report very small and not very systematic evaluations. (See Ritchie (2004, Chap. 10) for a fuller review of these systems.)

As our program is a punning riddle generator, JAPE (Section 3 below) and the systems reviewed above are its antecedents. Other work in computational humour has included a program which could construct amusing acronyms (Stock and Strapparava, 2003, 2005), a recogniser for basic "knock-knock" jokes (Taylor and Mazlack, 2004), a study of how machine-learning techniques could separate joke texts from non-jokes (Mihalcea and Strapparava, 2006), a very preliminary generator for insults based on 'scalar humour' (Binsted et al., 2003), and a program which, for a particular class of jokes, selects a punchline for a joke set-up (Stark et al., 2005). Although some of these were on a slightly larger scale than the pun generators described above, all of them were research prototypes, with no claims to be usable applications.

3 The JAPE riddle generator

The JAPE program (Binsted and Ritchie, 1994, 1997; Binsted, 1996) generated certain classes of punning riddles. Some of the better examples were the following:

(2) How is a nice girl like a sugary bird?
Each is a sweet chick.

(3) What is the difference between leaves and a car? One you brush and rake, the other you rush and brake

(4) What is the difference between a pretty glove and a silent cat? One is a cute mitten, the other is a mute kitten.

(5) What do you call a strange market? A bizarre bazaar.

JAPE used three types of symbolic rules (*schemas*, *description rules*, *templates*) to characterise the possible linguistic structures (Ritchie, 2003). Our variants of these mechanisms are described in detail below (Section 4).

JAPE stands out from the other early pun-generators in two respects: it used a large, general-purpose lexicon, WordNet (Miller et al., 1990; Fellbaum, 1998), rather than a small hand-crafted one, and a properly controlled evaluation of the output was carried out (Binsted et al., 1997). The latter study showed that JAPE-generated jokes were reliably distinguished from non-jokes, human-generated jokes were more often deemed to be jokes than JAPE-generated jokes, JAPE-generated jokes were funnier than non-jokes, and human-generated jokes were funnier than JAPE-generated jokes. Also, a JAPE output ((3) above) was rated the funniest in the data set.

As well as developing the mechanisms for punning riddle generation, Binsted suggested, in passing, the idea of using such a program for interactive language teaching. However, the JAPE implementation was still just a research prototype, and there were certain aspects which would have to be altered or rebuilt if it was to be used for practical purposes. These limitations were roughly in the areas of *usability* and *output quality*. To be more precise:

(i) The only behaviour of the program was to create riddles, one after another, with very few parameters available for variation. Also, these parameters were internal to the mechanism (e.g. the choice of schema) and might not make sense to an ordinary user.

(ii) The program worked by exhaustively searching for words and phrases which would match its schemas and templates. There was no way to guide the software (e.g. to make a joke on a particular topic).

(iii) There was no real user interface – the user (always a knowledgeable researcher) would invoke the program from a simple command interface.

(iv) The search for suitable words, being unintelligent and exhaustive, could (with a large lexicon) be very slow; Binsted's test runs took hours. Hence, the response time was useless for interaction with a user.

(v) The jokes were of very variable quality, with the proportion of intelligible jokes being quite small; the proportion of *good* intelligible jokes was very small.

(vi) Facilities for comparing words for similarity of sound were quite primitive. In particular, there was no provision for approximate matches (near-homophony) and correspondences between written and phonetic forms of words were slightly ad hoc.

Of these, (iii) and (iv) had to be remedied for our application, and (i) and (ii) were serious drawbacks. The more we could do about (v), the better, and addressing (vi) would contribute to this.

4 The joke generator

The STANDUP generator consists of three stages, as in JAPE; all are implemented in Java. Each stage consists of instantiating a particular kind of rule: *schemas* (Section 4.1), *description rules* (Section 4.2) – both supported by a large dictionary– and *templates* (Section 4.3).

4.1 Schemas

A schema consists of 5 parts:

Header: This attaches a symbolic name to the schema, and lists its parameters.

Lexical preconditions: This is a collection of constraints specifying the core items needed for a particular subclass of riddle. Items can be either *lexemes* (lexical entries) or *word forms* (the orthographic textual representation of a word). Constraints can involve syntactic categorisation (e.g. a lexeme is a noun), phonetic relations (e.g. two items rhyme), structural relations (e.g. an item X is a compound noun made up of components Y and Z), and semantic relations (e.g. one lexeme is a hypernym of another).

Question specification: This specifies how certain variables in the schema are, once instantiated, to be described within the question part of the eventual riddle. This, and the answer specification, supply the input to the description rules (Section 4.2 below).

Answer specification: This is like the **Question specification**, but contributes to the answer in the riddle.

Keywords: This lists the subset of the schema's variables which will be bound to lexemes. It is used to define a notion of *equivalence* between similar riddles: two riddles are deemed equivalent if they use the same schema with the same instantiation of the keyword variables. The generator tracks which instantiations have been used so far with a particular user, and does not offer 'equivalent' jokes again to the same user.

Informally, the schema's variables can be instantiated with values from the lexicon, providing they meet the constraints in the lexical preconditions.

There are 11 schemas (for 11 underlying kinds of joke). A typical schema is given in Figure 1. Following

```
Header: newelan2(NP, A, B, HomB)
Lexical preconditions:
  nouncompound(NP,A,B),
  homophone(B,HomB), noun(HomB)
Question specification:
  {shareproperties(NP, HomB)}
Answer specification: {phrase(A,HomB)}
Keywords: [NP, HomB]
```

Figure 1: A typical STANDUP schema

the practice of the JAPE system, relations and properties are expressed here in Prolog-style (logic-like) notation, with predicates applied to arguments. Although each schema was designed using this notation, in the actual implementation the lexical preconditions were compiled into an expression in the database query language SQL, to facilitate the finding of suitable variable values within the lexical database (implemented using the PostgreSQL software package[1]). This compilation was done by hand, although in principle the process could be fully automated.

[1]`http://www.postgresql.org`

The `newelan2` schema given above could have an instantiation in which `NP`= *computer screen*, `A` = *computer*, `B` = *screen*, and `HomB` = *scream* (the relation `homophone` means that the two items lie within the current threshold for phonetic similarity; i.e. "homophones" can be approximate). This could give rise (after two further phases of processing — Sections 4.2, 4.3 below) to a riddle such as (6):

(6) What do you call a shout with a window?
A computer scream.

The question specification and the answer specification show how the instantiating values have to be passed on to the next phase (Section 4.2 below), by embedding the relevant variables within symbolic expressions which act as signals about what is to be done with these values. In this example, the question specification would be `shareproperties(computer screen, scream)` and the answer specification would be `phrase(computer, scream)`.

4.2 Constructing descriptions

The middle phase of joke generation – constructing descriptions – was not in JAPE-1 (Binsted and Ritchie, 1994, 1997), but was introduced in JAPE-2 (Binsted, 1996; Binsted et al., 1997). It encodes possible linguistic variations, given core values from the schema instantiation.

The question specification and answer specification are handled separately. Each is matched non-deterministically against a set of description rules. These rules have a structure roughly similar to schemas, in that they have a *header*, some *preconditions*, and an output expression, the *template specifier* (Figure 2).

```
Header: shareproperties(X,Y)
Preconditions:
  meronym(X, MerX), synonym(Y, SynY)
Template specifier: [merHyp, MerX, SynY]
```

Figure 2: A sample description rule

In the example above, the question specification `shareproperties(computer screen, scream)` would match the header for the rule in Figure 2. This matching causes the data values (`computer screen, scream`) to be bound to the local variables `X,Y` of the rule, ready for the preconditions of the rule to be tested. These preconditions check further lexical properties and relations, to determine whether this rule is actually applicable in this case. (As with schema preconditions, the implementation represents these expressions in SQL, to facilitate searching of the lexical database.) This may involve testing for the existence of further values (e.g. values for `MerX`, `SynY` in the example above), thereby resulting in the binding of more variables (local to the rule). For example, starting from `X` = *computer screen* and `Y` = *scream*, the precondition testing might find (in the lexicon) variable values `MerX` = *window* and `SynY` = *shout*, thereby satisfying all the conjuncts of the precondition. If the precondition testing succeeds, the template specifier is instantiated (using the current variable values), and these expressions are passed on to the third stage of joke generation, *template filling*. Here, this expression would be `[merSyn, window, shout]`).

The answer specification `phrase(computer, scream)` matches a very basic description rule which passes both values on in an expression `[simple, computer, scream]`; this will later be interpreted (at the template phase) as a concatenation command.

However, the same schema instantiation could have led, if other values were found for `MerX` and `SynY` in the description rule used for the question, to a slightly different variant, such as (7).

(7) What do you call a cry that has pixels?
A computer scream.

Or if a different description rule had been chosen for the question specification, the joke might have been (8).

(8) What do you get when you cross a shout with a display?
A computer scream.

Here, the central idea of the joke (captured by the schema instantiation) is essentially the same in all three versions.

Hence, there are two distinct mechanisms used to achieve textual variation. The variation illustrated above involves slightly different phrases which originate from the same semantic material expanded and realised in varying ways. These are constructed by this middle phase, which is a non-humorous set of linguistic rules about how to build descriptive phrases. (These variations usually occur in the riddle's question, but the data for the answer is also passed through this middle stage, so as to have a cleaner architecture, and also to allow for possible minor adjustments being needed in the linguistic form of the answer data, which does occur with some joke types.)

On the other hand, different stereotyped joke-framing phrases, such as *What do you get when you cross* ____ or *What is the difference between* ____ are handled by the third phase (Section 4.3) below.

4.3 Surface templates

A template is, loosely speaking, a fixed string of text with some blank slots available for other textual material to be inserted. Building text by filling data (e.g. phrases) into a template is a long-standing and much-used approach to the generation of simple natural language text (Reiter and Dale, 2000). It lacks linguistic subtlety and can be inflexible, but it is very convenient when a particular application (as here) needs to build a few stereotyped forms of text which vary only in a few well-defined places. We have three types of template: *phrasal*, *question* and *answer*. Phrasal templates put the finishing touches to phrases built by the previous stage (description construction), for example inserting articles or prepositions as needed. A question template has the broad outline of a riddle question (e.g. *What do you call a* ____ *?*) with slots for phrases to be inserted, and an answer template has the broad structure of a riddle answer (e.g. *They're both* ____) with slots for phrases to be inserted.

A template has two parts: the *header* and the *body*. The expressions provided by the description rules, such as `[merSyn, window, shout]` and `[simple, computer, scream]` are non-deterministically matched against the headers of templates of the appropriate type (question or answer). This causes variables in the template header to be instantiated to the values (such as *window*, *shout*). These values are thereby passed into the body, which is a skeletal textual structure, such as `What do you call a NP(X,Y)`. Recursively, the template handler matches `NP(shout, window)` to a set of phrase templates, one of which yields *NP(shout) with a NP(window)*, and a further template match produces *a shout with a window*. The answer is also produced by the template module, but for an expression like `[simple, computer, scream]` there are no recursive calls – a single phrasal template produces *a computer scream*.

There are various restrictions about which question templates are compatible with which answer templates, and also which templates are viable for the values coming from a particular schema. These combinations are coded up in a table; these are known as the *joke types*, as we found it useful to characterise types of joke in terms of underlying rule combinations.

5 Going beyond JAPE

5.1 The lexicon

Using WordNet as its dictionary gave JAPE several benefits: many lexical entries (about 200,000), word-senses grouped into synonym sets, information about hyponym/hypernyms and meronyms; and the data is freely available in machine-manipulatable form. We therefore took WordNet as our starting point. However, before designing our system, we carried out consultations with relevant experts: adult users of software for augmentative and alternative communication (AAC), and speech/language therapists. This added further requirements to those needed just for joke generation.

5.1.1 Pictures

Our experts were adamant that children with limited literacy would need pictorial images to be displayed alongside words wherever possible. Preferably, these images should be familiar, and compatible with other uses of images that the children might have met. Fortunately, two companies who produce AAC software (Widgit Software Ltd and Mayer-Johnson LLC) kindly gave us permission to use their picture libraries. However, we had to expend a considerable amount of effort manually linking pictures to appropriate WordNet senses (not just to word forms).

5.1.2 Phonetic representation

In order to construct approximate puns (e.g. matching *rude* and *road*), we needed a representation of the phonetic form of each word (orthography, as in WordNet, can be misleading for punning). We used the Unisyn pronunciation dictionary[2] to add phonetic forms to more than 115,000 word forms in our lexicon. This allowed the implementation of a more subtle matching algorithm

[2]`http://www.cstr.ed.ac.uk/projects/unisyn`

for phonetic similarity (near-homophony), based on Ladefoged and Halle (1988) and minimum edit-cost.

5.1.3 Familiarity of words

It is essential to be able to restrict the available vocabulary to words which the intended users (young children, perhaps with poor literacy) are likely to know, as there will be no beneficial effect, and probably some demoralisation, if the software produces riddles with words which are totally incomprehensible. JAPE was liable to produce riddles using extremely obscure words, such as (9).

(9) What do you get when you cross a vitellus and a saddlery? A yolk yoke.

All the available sources of word-familiarity information manifested one (or more) of three problems: assigning ratings (e.g. corpus frequencies) to *word-forms*, not to *word-senses*; sparseness, i.e. covering only a few thousand words; unreliability or unsuitability for our purposes. To address this, we applied a hybrid strategy, which there is not space here to document. This involved assigning ratings of *priority* to several different resources and also scaling the ratings from each resource into a sub-interval of [0,1]. A word-sense was then assigned a rating (in [0,1]) by the highest priority source for which it had a rating.

The STANDUP joke generator has an (adjustable) filter on the ratings of the lexemes used. This can be used to limit the unfamiliarity of words used.

5.1.4 Vocabulary restriction

It must be possible to avoid the use of words which are highly unsuitable for the user population (e.g. swear words, sexual terminology). JAPE was quite capable of producing jokes which, while semantically valid, were socially unacceptable for our target audience; e.g. (10).

(10) What do you call a capable seed?
An able semen.

We introduced a *blacklist* which contains words that must not be used anywhere by the system. It was populated by searching the Shorter Oxford English Dictionary for all entries tagged as either *coarse slang* or *racially offensive*; a few further entries were added to this list by the project members based on personal knowledge of words likely to be deemed unsuitable by teachers. (Despite this, a teacher objected to one riddle with quite innocent lexemes: *What do you call a queer rabbit? A funny bunny.*)

5.2 Avoiding simple faults

Although the JAPE riddle generator produced structurally correct texts, some of them were far from acceptable as jokes. We implemented various heterogeneous improvements, generally formal checks to eliminate configurations of lexemes which would lead to (intuitively speaking) poorer output; that is, we did not so much positively improve the jokes as selectively close off some of the more noticeable and formally definable routes to weak jokes.

5.2.1 Shared roots.

Early versions of STANDUP produced riddles in which the same word (or morphological variants of a word) appeared in both the question and the answer, which tended to spoil the joke: *What do you get when you cross a school principal with a rule? A principal principle.* Using information from the Unisyn dictionary we were able to associate a 'root' field with lexemes, and filter out riddles in which the same root appeared in question and in answer.

5.2.2 Excessive abstraction.

Many words in our lexicon were ultimately linked (via the WordNet hyponym/hypernym hierarchy), to very abstract entries such as *entity* or *human activity*. This could cause riddles to be excessively obscure; for example: *What do you get when you cross an aristocracy with a quality? A nobility mobility.* Here, *quality* is a hypernym of *mobility*, but this gives an excessively imprecise question. We therefore placed some of the roots of the hypernym forest in a further list of lexemes to be excluded from use. This was done by subjective judgement of the degree of abstraction, not by considering jokes which included the concepts. Although this removed many baffling riddles, the phenomenon of unworkable abstraction is more subtle. Example (11) is from an early version of STANDUP (before we improved the phonetic matching), and presumably puns on *double-decker* (a two-level bus):

(11) What do you call a cross between a coach and a trained worker?
A double baker.

The phrase *trained worker* is found by a description rule seeking hypernyms of *baker*. But *trained worker*, although not as wildly abstract as *entity* or *quality*, is still too vague to invoke the specific notion of *baker*. A hypernym should be used in a riddle question only if it is close enough in meaning to the target item (here, *baker*). It is hard to specify an appropriate criterion of "closeness".

5.3 Changes in coverage

STANDUP's set of schemas is slightly different from that in JAPE. Although we added one further schema (so that substitutions of a word into another word could happen at the end as well as at the start), there were fewer schemas (11 to JAPE's 15). This is due to two factors. Firstly, we were able to combine certain JAPE schemas which were very similar. Secondly, we had to omit some of the JAPE schema, for jokes such as (3). These schemas rely on information about what nouns are suitable subjects or objects for verbs, which, in the JAPE project, was compiled by hand in a relatively labour-intensive fashion. It was not clear how best to scale this up automatically (although it is conceivable that "Word Sketch" data (Kilgarriff et al., 2004) might help). Given the limited resources of our project, we had to do without these schemas. This highlights the difference in purpose between a research prototype and a working system. A prototype is often built to test whether some particular algorithm or design will work in principle – a proof of concept. This can be achieved if the necessary knowledge resources or environment can be created, even if only on a small scale. Thus, Binsted demonstrated a valid computational route to riddles such as (3) (and one or two other verb-based types), but this is very different from devising a practical means to make a large scale system which exploits this route.

5.4 User interaction

Perhaps the most significant advance in the STANDUP system is interaction between user and joke generator. There is a bright, colourful child-friendly GUI, using specially-designed graphic images, which allows the user to control the generator through a number of buttons. Thus a joke can be requested which contains a specified word, is on a given topic (e.g. *animals*) or is of a given type (e.g. *where the start of words are swapped round*). The user can browse through past jokes made at previous sessions, or save jokes to his/her own 'favourites'. When choosing a word for a joke, the user can browse through the lexicon.

Response time ranges from under a second to several seconds. This has been achieved by efficient coding and by caching (as database tables) lexical information used by the joke generator, such as near-homophone lists, tuples of lexemes forming spoonerisms, etc., and also instantiations of schemas.

The software has a Control Panel through which a researcher, teacher or carer can customise the system's behaviour (what appears on the screen, what kinds of jokes are available, what input/output mechanisms are used, etc.) for individual users, in a very flexible manner.

5.5 Joke telling

Part of the motivation for this work came from the idea that a child who used a voice-output communication aid (VOCA) – i.e. using a speech synthesiser in order to "speak" – might like to incorporate jokes into their conversation. However, it would have been over-ambitious to attempt to incorporate the joke-building functionality into a VOCA at this stage of development, so we instead developed a stand-alone system which a child could experiment with. Our software had a built-in text-to-speech system (using FreeTTS[3]) for reading messages, button labels, etc. to the user. There was also a facility whereby the user could, having obtained a joke, "tell" it step-by-step (question, pause, answer) by getting the software to speak the text, with the user controlling this process through the pointing device. This proved to be highly popular with the users (Section 6 below), as the children could tell their newly-built jokes immediately without having to switch over to their VOCA and enter the text.

6 Evaluating the system

For our software, usability and effectiveness for our target group were central. We therefore evaluated STANDUP with a group of children with CCN (fuller details can be found elsewhere).

A single case-study methodology was used with nine pupils at a special-needs primary school. All had cerebral palsy, and were in the 8 - 13 year age group. Their literacy levels were rated as either *emerging* or *assisted*. Eight of the participants were users of various communication aids, and could interact with these via touch screens or, in four cases, head switches. Children were taken through five phases: *baseline testing, introductory training, intervention, evaluation, post-testing*, where the three central phases involved sessions with the software. *Introductory training* consisted of familiarisation with the system, aided by one of the project team. *Intervention* involved the child having a simple task (suggested by the researcher) to try with the software, such as finding a joke on a particular topic. The researcher also offered guidance as necessary. For the *evaluation* phase, tasks were again suggested, but no help was given unless absolutely essential. Sessions were video-taped for analysis, and the software logged user-interactions into a disk file. Follow-up interviews and questionnaires were conducted with school staff and the participants' parents.

In the *baseline testing*, two standard multiple-choice tests for facility with words were administered: Clinical Evaluation of Language Fundamentals, CELF, (Semel et al., 1995), in which 27 questions each ask for a choice of 2 semantically related words from a set of 4, and a rhyme-awareness test from the Preschool and Primary Inventory of Phonological Awareness, PIPA (Frederickson et al., 1997). We also tested each child's grasp of punning riddles, using the Keyword Manipulation Task (O'Mara, 2004), simply to check our assumptions about the level of the children's understanding.

The *post-testing* with PIPA (testing awareness of rhyme) showed no signs of improvement (although 6 of the 9 scored above 80% on both pre- and post-test, suggesting a possible ceiling effect). On the CELF post-test, all but one of the participants improved, the mean improvement being 4.1 out of 27 (paired t-test, two-tailed, yields $t = -3.742$, $df = 8$, $p = 0.006$). It is difficult to conduct randomised controlled trials in the AAC field, as the set of people who use AAC tends to be highly heterogeneous. In the absence of any comparison with a control group, it is hard to infer much from the scores.

All the children reacted very positively to their time with the STANDUP software. One of the older boys, who had good verbal abilities, complained about the quality of the jokes, but made insightful comments on possible improvements to the system. The pupils spontaneously used the software (some without need for prompting), enjoyed having the software tell the jokes to others, and re-told the jokes afterwards to parents and others. Children initiated interaction, some for the first time. This may be because they felt that the program provided them with novel language, and that they could truly control an interaction by telling a new joke, instead of repeating vocabulary stored in devices by their therapists/teachers. The computer-generated jokes became part of an existing class project, with pupils posting their favourite examples publicly.

Although this was a very small qualitative study, with no ambitions to show skill improvements over such a short term, there was anecdotal evidence (from parents and teachers) that children's attitudes to communication had improved. Since it was far from clear at the outset

[3] http://freetts.sourceforge.net/docs/index.php

of our project whether children with CCN would even be able to use the planned software, the results are not trivial.

7 Discussion

7.1 The outcome

We have designed and built a fully working, large-scale, robust, interactive, user-friendly riddle generator, with a number of auxiliary facilities such as speech output and adjustable user profiles, remedying the limitations of JAPE listed in Section 3. It can create millions of jokes, has been evaluated in a non-trivial way, and is available for download over the WWW. Although we started from the JAPE ideas, our design and implementation effort was probably around four to five person-years of full-time work (excluding the evaluation). The area where further improvement is most needed is joke quality.

7.2 Creativity

7.2.1 Is the software creative?

Binsted did not claim that JAPE was creative, but Boden (1998, 2003) discusses it as an example of a creative program. Given STANDUP's relationship to JAPE, the question of creativity again arises. Anecdotal evidence suggests that the program produces novel and acceptable jokes, but, in the absence of a formal evaluation of the output (cf. Binsted et al. (1997)), no solid claim can be made. All of these jokes are based on hand-crafted schemas, so there is no creation of novel *types* of joke. (In the sense of Boden (1998), such innovation would be *transformational* rather than *exploratory* creativity.)

Is STANDUP "more creative" than JAPE? The devices described in Section 5 above alter the set of output items (compared to JAPE's, or – more realistically – to an earlier version of STANDUP). The modifications in Section 5.2 eliminate poorer items, thereby enhancing the overall output quality. By the formal criteria 1 to 4 (and possibly 5) in Ritchie (2001a, forthcoming), the elimination of faulty items would improve the program's ratings, as these criteria assess the *proportion* of the output items which are categorisable as jokes, or which are classed as *good* jokes. Some other changes (Section 5.3) eliminate certain classes, making the output set less varied. Ritchie's proposed criteria do not assess output set variety, so this would not affect the rating of the STANDUP program, but Pereira et al. (2005) hint that less variety in output is a sign of lower creativity. It is hard to draw firm conclusions here (except perhaps that these criteria are insufficiently subtle for making fine comparisons of creativity).

7.2.2 Supporting creativity

Given that the whole project was intended to give support to the users' skills development, perhaps a relevant viewpoint to consider is the extent to which the software supports or encourages *human* creativity. This is very hard to assess. If there were a fuller study to determine the effect of software usage on a child's skills (including social actions), perhaps some educational tests of creative thinking could be used to assess this aspect.

7.3 Future directions

Further studies. It would be very illuminating to carry out a long term study of the use of the software by children, to obtain some idea of the effects such language play has on linguistic, communicative or social skills. Comparisons with other "language play" educational software would be interesting, as would studies with other user populations (e.g. children with autism, second-language learners).

Improving the system. Because of our requirements studies (and the limited time available), we implemented relatively simple facilities for user interaction. These could be extended, to allow greater participation by the user in the joke-building process. It would also be interesting to handle other joke types (e.g. 'knock-knock' jokes (Taylor and Mazlack, 2004)).

A testbed for humour. The STANDUP software could become a framework to test ideas about humour, in limited ways. Allowing users to record reactions to jokes would allow the collection of data about which generated items work best, a resource for researchers. Alternatively, it might be possible to embed, in a future version, some conjecture about factor(s) which affect funniness, and then determine the empirical effectiveness of this.

7.4 Conclusions

Computational humour may still be at a basic level, but the work here represents an significant milestone in its development: a complete working system that addresses a practical application and is accessible for ordinary users.

Acknowledgements

This work was supported by the UK's Engineering and Physical Sciences Research Council. The Widgit Rebus symbols are the property of Widgit Software Ltd and are used under licence. The Picture Communication Symbols are the property of Mayer-Johnson LLC and are used under licence. We are extremely grateful to Capability Scotland and the staff and pupils at Corseford School for their help with the evaluation sessions.

References

Binsted, K. (1996). *Machine humour: An implemented model of puns*. PhD thesis, University of Edinburgh, Edinburgh, Scotland.

Binsted, K., Bergen, B., and McKay, J. (2003). Pun and non-pun humour in second-language learning. In *Workshop Proceedings, CHI 2003*, Fort Lauderdale, Florida.

Binsted, K., Pain, H., and Ritchie, G. (1997). Children's evaluation of computer-generated punning riddles. *Pragmatics and Cognition*, 5(2):305–354.

Binsted, K. and Ritchie, G. (1994). An implemented model of punning riddles. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, Seattle, USA.

Binsted, K. and Ritchie, G. (1997). Computational rules for generating punning riddles. *Humor: International Journal of Humor Research*, 10(1):25–76.

Boden, M. A. (1998). Creativity and Artificial Intelligence. *Artificial Intelligence*, 103:347–356.

Boden, M. A. (2003). *The Creative Mind*. Routledge, London, 2nd edition. First edition 1990.

Fellbaum, C. (1998). *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, Mass.

Frederickson, N., Frith, U., and Reason, R. (1997). *The Phonological Assessment Battery*. NFER-Nelson, Windsor.

Hulstijn, J. and Nijholt, A., editors (1996). *Proceedings of the International Workshop on Computational Humor*, number 12 in Twente Workshops on Language Technology, Enschede, Netherlands. University of Twente.

Kilgarriff, A., Rychly, P., Smrz, P., and Tugwell, D. (2004). The Sketch Engine. In *Proceedings of EURALEX 2004*, pages 105–116, Lorient, France.

Ladefoged, P. and Halle, M. (1988). Some major features of the international phonetic alphabet. *Language*, 64(3):577–582.

Lessard, G. and Levison, M. (1992). Computational modelling of linguistic humour: Tom Swifties. In *ALLC/ACH Joint Annual Conference, Oxford*, pages 175–178.

Lessard, G. and Levison, M. (1993). Computational modelling of riddle strategies. In *ALLC/ACH Joint Annual Conference, Georgetown University, Washington, DC*, pages 120–122.

Levison, M. and Lessard, G. (1992). A system for natural language generation. *Computers and the Humanities*, 26:43–58.

Lindsay, G. and Dockrell, J. (2000). The behaviour and self-esteem of children with specific speech and language difficulties. *British Journal of Educational Psychology*, (70):583–601.

McKay, J. (2002). Generation of idiom-based witticisms to aid second language learning. In Stock et al. (2002), pages 77–87.

Mihalcea, R. and Strapparava, C. (2006). Learning to laugh (automatically): Computational models for humor recognition. *Computational Intelligence*, 22(2):126–142.

Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D., and Miller, K. (1990). Five papers on WordNet. *International Journal of Lexicography*, 3(4). Revised March 1993.

O'Mara, D. (2004). *Providing access to verbal humour play for children with severe language impairment*. PhD thesis, Applied Computing, University of Dundee, Dundee, Scotland.

Pereira, F. C., Mendes, M., Gervás, P., and Cardoso, A. (2005). Experiments with assessment of creative systems: an application of Ritchie's criteria. In Gervás, P., Veale, T., and Pease, A., editors, *Proceedings of the Workshop on Computational Creativity, 19th International Joint Conference on Artificial Intelligence*, volume 5-05 of *Technical Report*, pages 37–44. Departamento de Sistemas Informáticos y Programación, Universidad Complutense de Madrid.

Reiter, E. and Dale, R. (2000). *Building Natural Language Generation Systems*. Cambridge University Press, Cambridge, UK.

Ritchie, G. (2001a). Assessing creativity. In *Proceedings of the AISB Symposium on Artificial Intelligence and Creativity in Arts and Science*, pages 3–11, York, England.

Ritchie, G. (2001b). Current directions in computational humour. *Artificial Intelligence Review*, 16(2):119–135.

Ritchie, G. (2003). The JAPE riddle generator: technical specification. Informatics Research Report EDI-INF-RR-0158, School of Informatics, University of Edinburgh, Edinburgh.

Ritchie, G. (2004). *The Linguistic Analysis of Jokes*. Routledge, London.

Ritchie, G. (forthcoming). Some empirical criteria for attributing creativity to a computer program. *Minds and Machines*. To appear.

Semel, E., Wiig, E. H., and Secord, W. A. (1995). *Clinical Evaluation of Language Fundamentals 3*. The Psychological Corporation, San Antonio, Texas.

Smith, M. (2005). *Literacy and Augmentative and Alternative Communication*. Elsevier Academic Press, Burlington.

Stark, J., Binsted, K., and Bergen, B. (2005). Disjunctor selection for one-line jokes. In Maybury, M. T., Stock, O., and Wahlster, W., editors, *Proceedings of First International Conference on Intelligent Technologies for Interactive Entertainment*, volume 3814 of *Lecture Notes in Computer Science*, pages 174–182. Springer.

Stock, O. and Strapparava, C. (2003). HAHAcronym: Humorous agents for humorous acronyms. *Humor : International Journal of Humor Research*, 16(3):297–314.

Stock, O. and Strapparava, C. (2005). The act of creating humorous acronyms. *Applied Artificial Intelligence*, 19(2):137–151.

Stock, O., Strapparava, C., and Nijholt, A., editors (2002). *Proceedings of the April Fools' Day Workshop on Computational Humor*, number 20 in Twente Workshops on Language Technology, Enschede, Netherlands. University of Twente.

Taylor, J. M. and Mazlack, L. J. (2004). Computationally recognizing wordplay in jokes. In *Proceedings of Cognitive Science Conference*, pages 2166–2171, Stresa, Italy.

Venour, C. (1999). The computational generation of a class of puns. Master's thesis, Queen's University, Kingston, Ontario.

Waller, A. (2006). Communication access to conversational narrative. *Topics in Language Disorders*, 26(3):221–239.

Automatizing Two Creative Functions for Advertising

Carlo Strapparava and **Alessandro Valitutti** and **Oliviero Stock**
FBK-irst, I-38050, Povo, Trento, ITALY
{strappa, alvalitu, stock}@itc.it

Abstract

The creation of advertising messages is a deep process of creative writing production. As far as the textual content is concerned, there are not many computational tools (besides the usual dictionaries, thesauri or program for performing of simple wordplays) that help the copywriter activity. In this work we explore the use of natural language processing and text animation techniques for proposing solutions to advertising professionals and improving the quality of advertising messages. In the proposed system, we consider two steps: (i) the creative variation of familiar expressions, taking into account the affective content of the produced text, (ii) the automatic animation (semantically consistent with the affective text content) of the resulting expression, using kinetic typography techniques.

Keywords: Natural Language Processing, Affective Text, Lexical Semantics, WORDNET, Text Animation.

1 Introduction

In modern advertising practice, it is common of "creatives" to be recruited and hired in pairs formed by a copywriter and an art director. They work in a creative partnership to conceive, develop and produce effective advertisements. While the copywriter is mostly responsible for the textual content of the creative product, the art director focalizes efforts on the graphical presentation of the message. Advertising messages tend to be quite short but, at the same time, rich of emotional meaning and persuasive power. While computational tools are an essential complement in many creative activities, e.g. graphical design, there are few tools for creation of textual messages (except for the usual dictionaries, thesauri or programs for performing simple wordplays).

In this paper we explore the development of computational tools to improve the quality of advertising messages, reducing the development time and possibly opening up the way to a full automatization of the whole process of creative writing production. We combine some computational functionalities for the creation of advertising messages. In particular, we implemented a strategy that is articulated in two steps. The first consists of the selection and creative variation of familiar common sense expressions (e.g. proverbs, idioms, cliches, movie titles, famous citations, etc.). The second step consists of the presentation of the expression through an automated text animation, and it is based on the use of kinetic typography. As we will see, the text animation can be built semantically consistent with the emotion we want to convey.

1.1 Advertising Messages and Optimal Innovation

An advertising message induces in the recipient a positive (or negative) attitude toward the subject to advertise (Petty and Wegener, 1998), for example through the evocation of an appropriate emotion. Another mandatory characteristic of an advertisement is its memorability. These two aspects of an ad increase the probability to induce some wanted behaviours, for example the purchase of some product, the choice of a specific brand, or the click on some specific web link. In the last case, it is crucial to make the recipient curious about the subject referred by the URL. The best way to realize in an ad both attitude induction and memorability is the generation of surprise, generally based on creative constraints.

In order to develop a strategy for surprise induction, we considered an interesting property of pleasurable creative communication that was named by Rachel Giora as the *optimal innovation hypothesis* (2003). According to this assumption, when the novelty is in a complementary relation to salience (familiarity), it is "optimal" in the sense that it has an aesthetics value and "induces the most pleasing effect".

Therefore the simultaneous presence of novelty and familiarity makes the message potentially surprising, because this combination allows the recipient's mind to oscillate between what is known and what is different from usual. For this reasons, an advertising message must be original but, at the same time, connected to what is familiar (Pricken, 2002). Familiarity causes expectations, while novelty violates them, and finally surprise arises.

1.2 Familiar Expression Variation

With "familiar expression variation" we indicate an expression (sentence or phrase) that is obtained as a linguistic change (e.g. substitution of a word, morphological or phonetic variation, etc.) of an expression recognized as familiar by recipients (e.g. selected by some collection of proverbs, famous movie titles, etc.). In this work we limited the variation to word substitution.

Moreover, an ad has to own a semantic connection with some concept of the target topic. At the same time, it has to be semantically related with some emotion of a prefixed valence (e.g. positive emotion as `joy` or negative emotion as `fear`).

We combined all these constraints in a compatible way with the optimal innovation hypothesis. The "innovation" is provided by the semantic similarity with the target topic and with the emotion, and the "optimality" is guaranteed by the assonance (i.e. the old and the new word have to be assonant, e.g. rhymed).

We considered in this process some recent works in computational humor (e.g. (Stock and Strapparava, 2003)), in which incongruity theory is exploited to produce funny variations of given acronyms[1]. In this work we extend this approach, focussing on the affective load of lexicon for the variation production and generating automatically typographical animations that are coherent with the emotions we want to communicate.

The paper is structured as follows. In Section 2 we introduce the resources used in the system, in particular (i) WordNet-Affect, an extension of the WordNet database in which some affective labels are assigned to a number of synsets; (ii) an affective semantic similarity, based on a Latent Semantic Analysis, which gives us an indication of the affective weight of generic terms; (iii) databases of familiar expressions and assonance tools; and (iv) a kinetic typography scripting language used for the final sentence animation. Section 3 describes the algorithm to variate familiar expressions and Section 4 displays some examples. Conclusions and future works are reported in Section 5.

2 Resources

2.1 Affective Semantic Similarity

All words can potentially convey affective meaning. Each of them, even those more apparently neutral, can evoke pleasant or painful experiences, because of their semantic relation with emotional concepts. While some words have emotional meaning with respect to the individual story, for many others the affective power is part of the collective imagination (e.g. words "mum", "ghost", "war" etc.).

We are interested in this second group, because their affective meaning is part of common sense knowledge and can be detected in the linguistic usage. For this reason, we studied the use of words in textual productions, and in particular their co-occurrences with the words in which the affective meaning is explicit. As claimed by Ortony et al. (Ortony et al., 1987), we have to distinguish between words directly referring to emotional states (e.g. "fear", "cheerful") and those having only an indirect reference that depends on the context (e.g. words that indicate possible emotional causes as "killer" or emotional responses as "cry"). We call the former *direct affective words* and the latter *indirect affective words* (Strapparava et al., 2006).

In order to manage affective lexical meaning, we (i) organized the direct affective words and synsets inside WordNet-Affect, an affective lexical resource based on an extension of WordNet, and (ii) implemented a selection function (named *affective weight*) based on a semantic similarity mechanism automatically acquired in an unsupervised way from a large corpus of texts (100 millions of words), in order to individuate the indirect affective lexicon.

Applied to a concept (e.g. a WordNet synset) and an emotional category, this function returns a value representing the semantic affinity with that emotion. In this way it is possible to assign a value to the concept with respect to each emotional category, and eventually select the emotion with the highest value. Applied to a set of concepts that are semantically similar, this function selects subsets characterized by some given affective constraints (e.g. referring to a particular emotional category or valence).

As we will see, we are able to focus selectively on positive, negative, ambiguous or neutral types of emotions. For example, given "difficulty" as input term, the system suggests as related emotions: IDENTIFICATION, NEGATIVE-CONCERN, AMBIGUOUS-EXPECTATION, APATHY. Moreover, given an input word (e.g. "university") and the indication of an emotional valence (e.g. positive), the system suggests a set of related words through some positive emotional category (e.g. "professor" "scholarship" "achievement") found through the emotions ENTHUSIASM, SYMPATHY, DEVOTION, ENCOURAGEMENT.

This fine-grained affective lexicon selection can open up new possibilities in many applications that exploit verbal communication of emotions. For example, (Valitutti et al., 2005) exploited the semantic connection between a generic word and an emotion for the generation of affective evaluative predicates and sentences.

WordNet-Affect and the Emotional Categories. WordNet-Affect is an extension of the WordNet database (Fellbaum, 1998), including a subset of synsets suitable to represent affective concepts. Similarly to what was done for domain labels (Magnini and Cavaglià, 2000), one or more affective labels (*a-labels*) are assigned to a number of WordNet synsets. In particular, the affective concepts representing an emotional state are individuated by synsets marked with the a-label EMOTION. There are also other a-labels for those concepts representing moods, situations eliciting emotions, or emotional responses. WordNet-Affect is freely available for research purpose at `http://wndomains.itc.it`. See (Strapparava and Valitutti, 2004) for a complete description of the resource.

We extended WordNet-Affect with a set of additional a-labels (i.e. the *emotional categories*), hierar-

[1] As far as computational humor is concerned and in particular funny variations of existing expressions, it is worth mentioning the work on pun creation of Binsted and Ritchie (1997).

	# Synsets	*# Words*	*# Senses*
Nouns	280	539	564
Adjectives	342	601	951
Verbs	142	294	430
Adverbs	154	203	270
Total	918	1637	2215

Table 1: Number of elements in the emotional hierarchy.

chically organized, in order to specialize synsets with a-label EMOTION. In a second stage, we introduced some modifications, in order to distinguish synsets according to emotional valence. We defined four additional a-labels: POSITIVE, NEGATIVE, AMBIGUOUS, NEUTRAL. The first one corresponds to "positive emotions", defined as emotional states characterized by the presence of positive edonic signals (or pleasure). It includes synsets such as `joy#1` or `enthusiasm#1`. Similarly the NEGATIVE a-label identifies "negative emotions" characterized by negative edonic signals (or pain), for example `anger#1` or `sadness#1`. Synsets representing affective states whose valence depends on semantic context (e.g. `surprise#1`) were marked with the tag AMBIGUOUS. Finally, synsets referring to mental states that are generally considered affective but are not characterized by valence, were marked with the tag NEUTRAL.

Computing Lexical Affective Semantic Similarity. There is an active research direction in the NLP field about sentiment analysis and recognition of semantic orientation from texts (e.g. (Turney and Littman, 2003; Liu et al., 2003; Mihalcea and Liu, 2006)). In our opinion, a crucial issue is to have a mechanism for evaluating the semantic similarity among generic terms and affective lexical concepts. To this aim we estimated term similarity from a large scale corpus. In particular we implemented a variation of Latent Semantic Analysis (LSA) in order to obtain a vector representation for words, texts and synsets.

In LSA (Deerwester et al., 1990), second order relations among terms and documents of the corpus are captured by means of a dimensionality reduction operated by a Singular Value Decomposition (SVD) on the term-by-document matrix. For the experiments reported in this paper, we run the SVD operation on the full British National Corpus[2].

SVD is a well-known operation in linear algebra, which can be applied to any rectangular matrix in order to find correlations among its rows and columns. SVD decomposes the term-by-document matrix $\mathbf{T}$ into three matrices $\mathbf{T} = \mathbf{U}\mathbf{\Sigma_k}\mathbf{V}^T$ where $\mathbf{\Sigma_k}$ is the diagonal $k \times k$ matrix containing the k singular values of $\mathbf{T}$, $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_k$, and $\mathbf{U}$ and $\mathbf{V}$ are column-orthogonal matrices. When the three matrices are multiplied together the original term-by-document matrix is re-composed. Typically we can choose $k' \ll k$ obtaining the approximation $\mathbf{T} \simeq \mathbf{U}\mathbf{\Sigma_{k'}}\mathbf{V}^T$. More specifically, in the experiments for this paper we use the matrix $\mathbf{T}' = \mathbf{U}\mathbf{\Sigma_{k'}}$, whose rows represent the term vectors in the reduced space, taking into account the first 100 dimensions (i.e. $k' = 100$).

[2]The British National Corpus is a very large (over 100 million words) corpus of modern English, both spoken and written (see `http://www.hcu.ox.ac.uk/bnc/`).

LSA can be viewed as a way to overcome some of the drawbacks of the standard vector space model (sparseness and high dimensionality). In fact, the LSA similarity is computed in a lower dimensional space, in which second-order relations among terms and texts are exploited. The similarity in the resulting vector space can be measured with the standard cosine similarity. Note also that LSA yields a vector space model that allows for a *homogeneous* representation (and hence comparison) of words, word sets, sentences and texts.

For representing word sets and texts by means of a LSA vector, we used a variation of the *pseudo-document* methodology described in (Berry, 1992). This variation takes into account also a *tf-idf* weighting schema (see (Gliozzo and Strapparava, 2005) for more details). In practice, each document can be represented in the LSA space by summing up the normalized LSA vectors of all the terms contained in it. Also a synset in WORDNET (and then an emotional category) can be represented in the LSA space, performing the pseudo-document technique on all the words contained in the synset. Thus it is possible to have a vectorial representation of each emotional category in the LSA space (i.e. the *emotional vectors*), and consequently we can compute a similarity measure among terms and affective categories. We defined the *affective weight* as the similarity value between an emotional vector and an input term vector (e.g. we can check how a generic term is similar to a given emotion).

For example, the noun "gift" is highly related to the emotional categories: LOVE (with positive valence), COMPASSION (with negative valence), SURPRISE (with ambiguous valence), and INDIFFERENCE (with neutral valence).

In summary, the vectorial representation in the Latent Semantic Space allows us to represent, in a *uniform* way, emotional categories, generic terms and concepts (synsets), and eventually full sentences.

2.2 Database of Familiar Expressions

The base for the strategy of "familiar expression variation" is the availability of a set of expressions that are recognized as familiar by English speakers.

We considered three types of familiar expressions: proverbs, movie titles, clichés. We collected 1836 familiar expressions from the Web, organized in three types: common use proverbs (628), famous movie titles (290), and clichés (918). Proverbs were retrieved in some of many web sites in which they are grouped (e.g. `http://www.francesfarmersrevenge.com/stuff/proverbs.htm` or `www.manythings.org/proverbs`). We considered only proverbs of common use. In a similar way we collected clichés, that are sentences whose overuse often makes them humorous (e.g. home sweet home, I am playing my own game). Finally, movie titles were selected from the Internet Movie Database (`www.imdb.com`). In particular, we considered the list of the best movies in all sorts of categories based on votes from users.

The list of familiar expressions is composed mostly of sentences (in particular, proverbs and clichés), but part of them are phrases (in particular, movie title list includes a significant number of noun phrases)

2.3 Assonance Tool

To cope with this aspect we got and re-organized the CMU pronouncing dictionary (`http://www.speech.cs.cmu.edu/cgi-bin/cmudict`) with a suitable indexing. The CMU Pronouncing Dictionary is a machine-readable pronunciation dictionary for North American English that contains over 125,000 words and their transcriptions.

Its format is particularly useful for speech recognition and synthesis, as it has mappings from words to their pronunciations in the given phoneme set. The current phoneme set contains 39 phonemes; vowels may carry lexical stress.

2.4 Kinetic Typography Scripting Language

Kinetic typography is the technology of text animation, i.e. text that uses movement or other changes over time. The advantage of kinetic typography consists in a further communicative dimension, combining verbal and visual communication, and providing opportunities to enrich the expressiveness of static texts. According to (Lee et al., 2002), kinetic typography can be used for three different communicative goals: capturing and directing attention of recipients, creating characters, and expressing emotions. A possible way of animating a text is mimicking the typical movement of humans when they express the content of the text (e.g. “Hi” with a jumping motion mimics exaggerated body motion of humans when they are really glad).

We explore the idea to have a link between lexical semantics of texts (automatically discerned through NLP techniques) and some kinetic properties exploited for animating the words. In this paper, we consider affective connotation of texts by exploiting the affective semantic similarity introduced above. This holds particularly for “indirect affective words” (Strapparava et al., 2006). For example, these words may indicate possible emotional causes (e.g. “monster”) or emotional responses (e.g. “cry”). Thus kinetic typography allows us to make the indirect affective meaning explicit in order to automatically augment the affective expressiveness of texts.

A first step was the individuation of an appropriate tool for the authoring and visualization of text animations. In particular, we wanted to act in an environment that allows us to realize animations in a very simple manner and to represent them in an easily exportable format. Functionalities for the automated composition of animations were our specific concern. To this aim we considered the Kinetic Typography Engine (KTE), a Java package developed at the Design School of Carnegie Mellon University (Lee et al., 2002). It allows us to create a potentially wide range of animations. Taking this engine as a starting point, we first realized a development environment for the creation and the visualization of text animations. Our model for the animation representation is a bit simpler than the KTE model. The central assumption consists of the representation of the animation as a composition of elementary animations (e.g. linear, sinusoidal or exponential variation). In particular, we consider only one operator for the identification of elementary animations (K-BASE) and three composition operators: kinetic addition (K-ADD), kinetic concatenation (K-JOIN), and kinetic loop (K-LOOP).

The K-BASE operator selects an elementary animation (named *elementary kinetic behavior*) as a temporal variation of some kinetic property. Elementary kinetic behaviors correspond to a subset of dynamic variations implemented in KTE, for example linear variation (*linear*), sinusoidal variation (*oscillate*), and exponential variation (*exponential*).

`linear`	linear variation
`oscillate`	sinusoidal variation
`pulse`	impulse
`jitter`	sort of “chaotic” vibration
`curve`	parabolic variation
`hop`	parabolic variation with small impulses at the endpoints
`hop-secondary`	derivative of hop, used as secondary effect to simulate elastic movements

Table 2: Some elementary kinetic behaviors

The kinetic addition (K-ADD) of two animations with the same start time is obtained by adding, for each kinetic property of text, the corresponding dynamical variation of each single animation. The kinetic concatenation (K-JOIN) consists in the temporal shifting of the second animation, so that the ending time of the first is the starting time of the second. The kinetic loop (K-LOOP) concatenates an animation with itself a fixed number of times. In the development environment it is possible to freely apply these operators for the real time building of new animations. Compositional structure of animations can be represented in XML format and then easily exported. Finally, an interpreter allows us to generate in real time the animation starting from its structural representation.

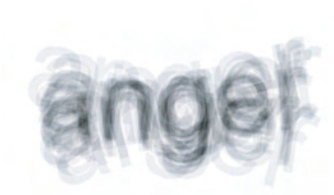


Figure 2: Jittering *anger*

After building the development tool, we selected a set of emotional categories and, for each of them, we created the corresponding text animations.

In particular, we focused on five emotional categories: joy, fear, surprise, anger, sadness (i.e. a subset of Ekman emotions (Ekman, 1977)).

The kinetic animation to associate to a fixed emotion

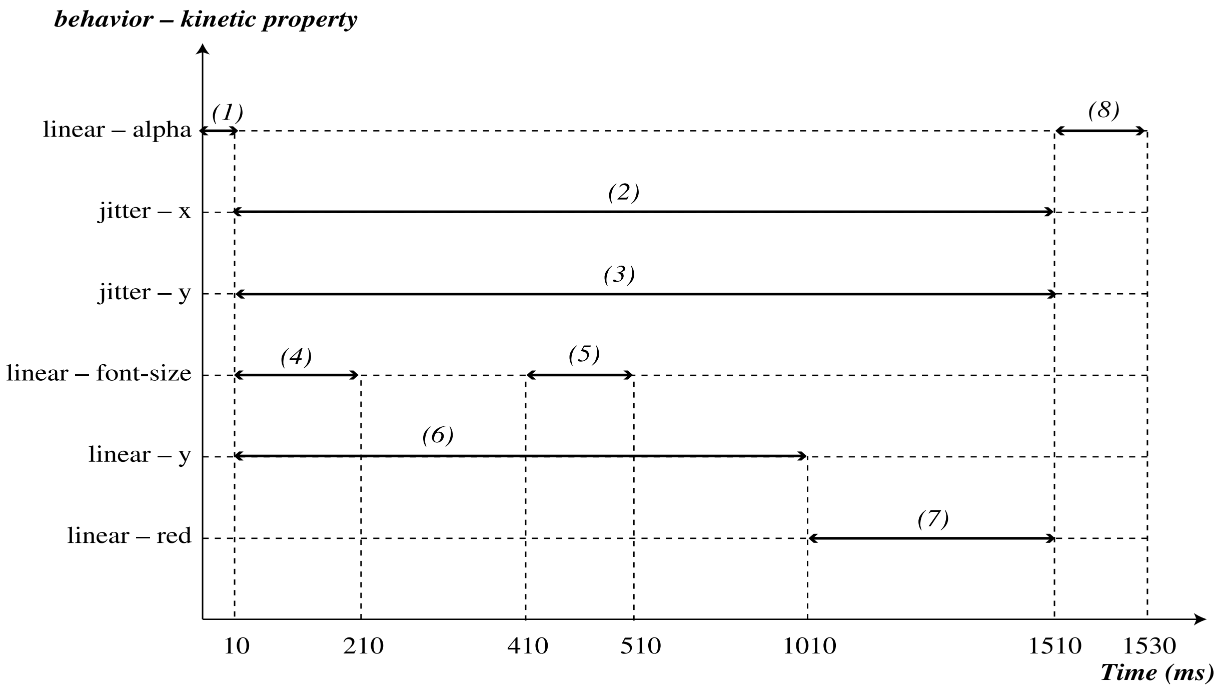


Figure 1: Kinetic behavior description for "anger" emotion

can be realized imitating either emotional and physiological responses (*analogous motion* technique), or tone of voice. We consider only animations of the first type, i.e. we represent each emotion with an animation that simulates a particular emotional behavior. In particular, JOY is represented with a sequence of hops, FEAR with palpitations, ANGER with a strong tremble and blush, SURPRISE with a sudden swelling of text, and finally SADNESS with text deflation and getting squashed. Thus we annotated the corresponding emotional categories in WORDNET-AFFECT with these kinematic properties.

Figure 1 displays in detail the behavior of the anger emotion, showing the time-dependent composition graph of the basic animations. The string appears (1) and disappears (8) with a linear variation of the alpha property (that defines the transparency of a color and can be represented by a float value). The animation is contained between these two intervals and its duration is 1500 ms. The first component is a tiny random variation of the position (2) (3), represented by x and y kinetic properties, with jitter behavior. The second component consists of an expansion of the string (4) and a subsequent compression (5). The third component is given by a slow rise up (6). The last component, before disappearing, is a color change to red (7). The whole behavior is then described and implemented using the scripting language introduced above.

As it is difficult to enjoy the animations on *static* paper, please visit the web page `http://tcc.itc.it/people/strapparava/affective-KT` where some downloadable short movies are available.

3 Algorithm

In this section, we describe the algorithm developed to perform a creative variation of an existing expression.

1. **Insertion of an input concept.** The first step of the procedure consists of the insertion of an input concept. This is represented by one or more words, a set of synonyms, or a WordNet synset. In the latter case, it is individuated through a word, the part of speech (noun, adjective, verb, or adverb), and the sense number, and it corresponds to a set of synonyms. Using the pseudo-document representation technique described above, the input word list is represented as a vector (named input-vector) in the LSA vectorial space.

2. **Generation of the target-list.** A list of terms (named target list) that are semantically connected (in the LSA space) with the input concept is generated. This target list represents a semantic domain that includes the input concept.

3. **Association of assonant words.** For each word of the target-list one or more possible *assonant words* are found. Then a list of word pairs (named *variation-pairs*) is created. Each pair has a *target word* as first element and an *assonant-word* as second element. At this point, the list of variation-pairs is filtered according to some constraints. The first one is syntactic (target-word and assonant-word must have the same part of speech). The second one is semantic (assonant word must not be included in the target-list), and its function is to maximize the probability to realize a semantic opposition between the elements of a variation pair. Finally, to each variation pair is associated an *emotion-label* (representing

the emotional category most semantically similar to the assonant word) with the corresponding value of affective weight. If a target word has more possible assonant words, we selected only that one having higher value of affective weight.

4. **Creative variation of familiar expressions.** In this step, the procedure gets as input a set of familiar expressions (in particular, proverbs and movie titles) and, for each of them, generate all possible variations. If an expression includes a word that is an element of at least one of the variation-pairs, then that word is substituted by the other element of the same pair.

5. **Ordering of familiar expressions.** The list of varied expressions is ordered according to the value of affective weight associated to the assonant word. The dimension of the input set of familiar expressions is crucial because it is related to the probability of generating a satisfactory creative variation.

6. **Text animation.** Finally, the varied expression is animated with kinetic typography technique. In particular, the assonant-word is animated according to the underlying emotion to emphasize the affective connotation.

4 Examples

In this section we want to show some examples of the creative function developed in our work and how it is useful for creating advertisements.

Simple creative variations. Using the affective weight function, it is possible to select a variation according to the valence (e.g. the substitution of the word *bad*, detected as negative, with *glad*, recognized as positive) or to some wanted affective direction. In Table 3 there are three variation of a movie title, according to three different emotions, to show that we can constrain the word substitution toward a word semantically similar to the desired emotional category.

Original	**Variation**	**Category**
Notting Hill	Notting *Thrill*	`Exhilaration`
	Notting *Still*	`Calmness`
	Notting *Chill*	`Gladness`

Table 3: Simple variation of a movie title

Humorous effects. Table 4 shows how word substitution may propagate the change of connotation at the level of the entire expression, and may also produce humorous effects. In particular, we observed that the semantic opposition, determined by switching affective polarity, generates another more complex semantic opposition at phrase (or sentence) level. In a possible scenario in which the creative user interacts with the system to generate creative expressions, the human recognition of high level humorous effects may be part of the creative interaction. The system proposes a list of possible candidates and the user makes the ultimate decision, selecting the creative variations that seem more meaningful[3].

Original	**Variation**	**Category**
when all else fails, read the instructions	when all else fails, *dread* the instructions	`Fear`
children and fools tell the truth	children and fools *repel* the truth	`Repugnance`
divide and rule	divide and *cool*	`Coolness`
a guilty conscience feels continual fear	a guilty conscience feels continual *cheer*	`Cheerfulness`

Table 4: Humorous variations

Advertising. In Table 5 there are some examples of automatically generated advertising messages. The creative variation has a semantic connection with a target topic and it is suitable for advertising purposes. In the first example, the original word *park* is substituted by the work *dark*, that have high semantic similarity with a target topic (*clothes*) and has a negative affective weight. The global expression communicates the idea that the colours for the new fashion must be clear and the dark clothes are old fashioned. The second example shows the substitution of the original word *night* with the word *fright* , that is semantically similar to the target topic *crash* and has a negative affective weight. The entire phrase can be used to warn young drivers about alcohol related driving accidents.

Original	**Variation**	**Category**
Jurassic Park	Jurassic *Dark*	`Gloom`
Saturday Night Fever	Saturday *Fright* Fever	`Fear`

Table 5: Variations for advertising messages

5 Conclusions

Exploiting some state-of-the-art natural language processing techniques, we described a system that produces creative variations of familiar expressions and animates them accordingly to the affective content. The creative textual variations are based on lexical semantics techniques such as affective similarity, while the animation makes use of a kinetic typography dynamic scripting language.

From an applied point of view, we believe that a thorough environment for proposing solutions to advertising

[3]In future work we are interested to refine a computational model that suggests the best semantic opposition or *incongruity* for humorous effect generation. Some useful considerations about the issue of incongruity can be found in (Ritchie, 1999; Veale, 2004)

professionals can be a practical development of this work, for the moment leaving the last word to the human professional. In the future, the potential of fully automatic production will find a big opportunity if advertisements are to be linked to an evolving context, such as incoming news, or changing of location of the audience, until a full user personalization of advertisements.

References

Berry, M. (1992). Large-scale sparse singular value computations. *International Journal of Supercomputer Applications*, 6(1):13–49.

Binsted, K. and Ritchie, G. (1997). Computational rules for punning riddles. *Humor*, 10(1).

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Ekman, P. (1977). Biological and cultural contributions to body and facial movement. In Blacking, J., editor, *Anthropology of the Body*, pages 34–84. Academic Press, London.

Fellbaum, C. (1998). *WordNet. An Electronic Lexical Database*. The MIT Press.

Giora, R. (2003). *On Our Mind: Salience, Context and Figurative Language*. Oxford University Press, New York.

Gliozzo, A. and Strapparava, C. (2005). Domains kernels for text categorization. In *Proc. of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, Ann Arbor.

Lee, J., Forlizzi, J., and Hudson, S. (2002). The kinetic typography engine: An extensible system for animating expressive text. In *Proc. of ACM UIST 2002 Conference*.

Liu, H., Lieberman, H., and Selker, T. (2003). A model of textual affect sensing using real-world knowledge. In *Proc. of the Seventh International Conference on Intelligent User Interfaces (IUI 2003)*, Miami.

Magnini, B. and Cavaglià, G. (2000). Integrating subject field codes into wordnet. In *Proc. of the 2^{nd} International Conference on Language Resources and Evaluation (LREC2000)*, Athens, Greece.

Mihalcea, R. and Liu, H. (2006). A corpus-based approach to finding happiness. In *Proc. of Computational approaches for analysis of weblogs, AAAI Spring Symposium 2006*, Stanford.

Ortony, A., Clore, G. L., and Foss, M. A. (1987). The psychological foundations of the affective lexicon. *Journal of Personality and Social Psychology*, 53:751–766.

Petty, R. and Wegener, D. (1998). Attitude change: Multiple roles for persuasion variables. In Gilbert, D., Fiske, S., and Lindzey, G., editors, *The handbook of social psychology*, pages 323–390. McGraw-Hill, New York, 4^{th} edition.

Pricken, M. (2002). *Creative Advertising*. Thames & Hudson.

Ritchie, G. (1999). Developing the incongruity-resolution theory. In *Proceedings of the AISB Symposium on Creative Language: Stories and Humour*, Edinburgh.

Stock, O. and Strapparava, C. (2003). Getting serious about the development of computational humour. In *Proceedings of the 8^{th} International Joint Conference on Artificial Intelligence (IJCAI-03)*, Acapulco, Mexico.

Strapparava, C. and Valitutti, A. (2004). WordNet-Affect: an affective extension of WordNet. In *Proc. of 4^{th} International Conference on Language Resources and Evaluation (LREC 2004)*, Lisbon.

Strapparava, C., Valitutti, A., and Stock, O. (2006). The affective weight of lexicon. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC 2006)*, Genoa, Italy.

Turney, P. and Littman, M. (2003). Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transactions on Information Systems (TOIS)*, 21(4):315–346.

Valitutti, A., Strapparava, C., and Stock, O. (2005). Lexical resources and semantic similarity for affective evaluative expressions generation. In *Proc. of the First International Conference on Affective Computing & Intelligent Interaction (ACII 2005)*, Beijing, China.

Veale, T. (2004). Incongruity in humor: Root-cause or epiphenomenon? *The International Journal of Humor*, 17(4).

Sessions 5 & 6
Frameworks for Creativity

Algorithmic Information Theory and Novelty Generation

Simon McGregor
Centre for Research in Cognitive Science
University of Sussex, UK
sm66@sussex.ac.uk

Abstract

This paper discusses some of the possible contributions of algorithmic information theory, and in particular the central notion of *data compression*, to a theoretical exposition of computational creativity and novelty generation. I note that the formalised concepts of pattern and randomness due to algorithmic information theory are relevant to computer creativity, briefly discuss the role of compression in machine learning theory and present a general model for generative algorithms which turns out to be instantiated by decompression in a lossy compression scheme. I also investigate the concept of novelty using information-theoretic tools and show that a purely "impersonal" formal notion of novelty is inadequate; novelty must be defined by reference to an observer with particular perceptual abilities.

1 Compression, Randomness and Pattern

The intuitive concepts of *pattern* and its converse, *randomness*, are of interest to those in the field of computer creativity. These concepts have been extensively explored in *statistical inference theory*: clearly, anything which has no pattern cannot be predicted; on the other hand, identifying a nonrandom pattern in data should allow us to predict it better than chance in future. Perhaps surprisingly, it turns out that the field of computer science known as *algorithmic information theory* has direct application to formalising the idea of randomness in observed data.

The concept of *algorithmic entropy* or *Kolmogorov* or *Kolmogorov-Chaitin* complexity is central to algorithmic information theory. The Kolmogorov complexity of a binary string is defined simply as the length of the shortest computer program which produces that string as an output. Some strings are *compressible*, i.e. there exists some computer program shorter than the string itself which produces that string as an output. For instance, the first 100,000 digits in the binary expansion of π can be generated by a program far shorter than 100,000 bits. A string consisting of the binary digit 1 repeated 1,000 times can be generated by a program shorter than 1,000 bits. However, it can be shown (Li and Vitanyi, 1997) that most strings are *incompressible*, i.e. they cannot be generated by a program shorter than themselves. Consequently, if you flip a perfectly random coin 100,000 times, the likelihood is that the sequence of heads and tails you obtain cannot be described by a program shorter than 100,000 bits. In algorithmic information theory a string is described as random if and only if it is incompressible.

Note that randomness in algorithmic information theory applies to *strings*, and not to the physical processes which generate those strings. A biased probabilistic random process such as radioactive decay could produce a sequence of 1s and 0s in which 1s were extremely common and 0s extremely rare; that sequence would be algorithmically nonrandom (because favouring 1s is a pattern) despite the fact that it was the product of a random process. Algorithmic randomness refers simply to the absence of pattern in a string.

Despite the best attempts of mathematicians to date, there are still some formal issues which restrict the usefulness of algorithmic entropy as an "objective" measure of randomness. Firstly, algorithmic entropy is provably uncomputable, so it cannot be used in practice. Secondly, in principle its exact value is dependent on the arbitrary reference machine on which programs are run, so that it is "non-arbitrarily" well-defined only in the asymptotic limit.

2 Lossy Compression

As mentioned above, most binary strings are incompressible. This means that theoretically, a compression program which allows objects to be reconstructed from their compressed representations cannot on average turn its inputs into shorter strings! Compression algorithms such as LZW[1] take advantage of the fact that some inputs (e.g. those containing many repeated substrings) are more likely in practice than others; for the majority of possible inputs (those not encountered in practice), the compressed representation will be longer than the original.

[1]Or the other algorithms used by compression utilities such as WinZip.

To make a compression program useful, one needs a *decompressor* (in practice, these two programs are usually bundled together). The decompressor takes the compressed representation of a string as its input and outputs the original string.

A *lossy compressor* is a program which destroys some information about its input in order to be able to produce (typically) shorter representations. For instance, the image compression standard JPEG is a lossy compression scheme. That is to say, when the output of a JPEG compression program is run through a JPEG decompressor, the result is typically not identical to the original input.

3 Generativity and Compression

Imagine a terminating computer program which is supposed to produce objects in our generative domain of interest (e.g. English poems, pictures of animals or rockabilly music). Due to the nature of digital computer programs, the objects generated must be encoded as binary strings (e.g. some ASCII text, or a PNG image, or an MP3 music file). Now, if the same program produces one object whenever it is run, and is capable of producing different objects, in formal terms the program can be considered as taking an input which determines its output. This conceptualisation is general enough to cover programs which operate in some random fashion (the random numbers can be provided as input). It is also general enough to encompass a non-terminating program which generates an infinite sequence of objects by changing its state: we can always write a terminating program which takes a number n as input and outputs the non-terminating program's nth generated object.

This leads to a view in which a generative computer model for a generative domain is seen as a program P which takes an arbitrary binary string as input and outputs a binary string encoding an object. Each bit of the input can be interpreted as a choice point in the process which generates the output. We'll presume that P is written in such a way that it cannot produce an "illegal" output no matter what the input is.

Let's additionally assume that it is possible to write an inverse program P', such that $P'(P(X)) = X$ always. The new program takes the encoding of an object Y and outputs a binary string X which can be fed into P (if there is such a string) to generate Y. If there is no input X which generates Y under P, P' finds the closest object Y^* to Y which can be produced by P, and outputs a binary string X^* which generates Y^* under P.

If its inputs are typically shorter than its outputs, the program P' as just defined is a standard *lossy compressor* program, and our generative model P is just the corresponding *decompressor*. In other words, a successful compression scheme which is computably decompressable yields a generative algorithm. Since optimal compression effectively abstracts away any pattern in data, this should not be surprising. The relation between lossy compression and generativity was noted as early as 1994 in the jokey paper Witten et al. (1994).

4 Learning and Compression

It is a well-known result in machine learning (Li and Vitanyi, 1997) that the shortest program which can produce observed data tends to generalise well to unseen data[2]. This is the centuries-old principle of *Occam's razor* - the simplest explanation is usually the best one. Any machine learning algorithm which is meant to generalise to unseen data from observed data must effectively perform some sort of compression.

Hence, a generative algorithm which is required to learn from its successes and failures can also be understood in terms of compression and algorithmic information theory. The most effective generalisation from past experience will in general be the one which compresses most, i.e. captures the pattern to the greatest possible extent.

5 Aesthetics and Compression

Unlike previous research, (e.g. Svangard and Nordin (2004); Schmidhuber (1997)) this paper does not consider the relation between compression and aesthetics. It focuses on learning, novelty and generativity, which are relevant both in artistic and non-artistic (for instance, engineering or mathematical) creative domains.

6 Novelty and Compression

6.1 The Problem of Novelty

Most theoretical accounts of creativity agree that creative products must be *novel*. Put simply, a product is novel if it is different from some set of already-observed things. Depending on the purpose, this reference set may be defined by what the originator has observed (what Boden (2003) calls personal- or p- creativity), or by what the entire historical community has observed (what Boden calls historical- or h- creativity). But we need to be careful here. "Different" does not merely mean non-identical. If I change one word of A. S. Byatt's "Possession", the resulting product is not novel[3] even though it is not identical to any pre-existing object. It is not "different enough" from prior works to qualify as "genuinely" novel.

That "different enough" is revealing: difference lies on a continuum, with identical objects being zero-different and other pairs of objects varying from hardly different to extremely different. Consequently, new products exhibit degrees of novelty, rather than falling into a binary novel / non-novel categorisation. The degree of novelty of a product depends on a (usually implicit) measure of *similarity* to a (usually implicit) reference class of pre-existing objects. For instance, in Saunders (2001), novelty is appraised using an implicit measure of similarity based on learning in unsupervised neural networks. In other words, novelty is relative not only to what has been seen before but also relative to how things are conceptually grouped together. For any formal version of novelty which relies

[2]Provided that the unobserved data comes from the same distribution as the observed data and that the distribution is computable.

[3]It is of course still *a* novel.

on similarity, it is necessary to specify what measure of similarity is being used.

6.2 Compression

There is a natural, impersonal formal sense in which two binary strings X and Y can be considered similar. The *information distance* (Bennett et al., 1998) tells us how close the algorithmic information in the two strings is. This distance, which is a metric up to an additive constant term, is defined as the length of the shortest program which produces Y given X as input and vice versa. In Bennett et al. (1998), it is described as a *universal cognitive similarity metric*. Formally,

$$E_1(X, Y) = \max\{K(X|Y), K(Y|X)\}$$

where $K(X|Y)$ is the conditional Kolmogorov complexity of X given Y (the length of the shortest progam which produces X given Y as input).

Although Kolmogorov complexity *per se* is uncomputable, an approximation to information distance has been successfully used in Cilibrasi and Vitanyi (2005) to identify similarity between sections of English text, similarity between DNA strings and similarity between musical melodies.

We could extend this formalism to give us measures of how "objectively" novel a binary string X_{n+1} is in comparison to previously known strings $X_1 \cdots X_n$. For instance,

$$\text{Nov}_1(X_{n+1}) = \min\{E_1(X_{n+1}, X_1), \cdots, E_1(X_{n+1}, X_n)\}$$

is the information distance from the new string to the most similar previously known string.

As we will see in the next section, however, the use of this "objective" similarity measure would be at fundamental odds with the goals and methods of computational creativity.

7 Tensions in "Objective" Novelty Generation

By definition, if novelty were held to be algorithmic randomness with respect to known previous examples, then there could not be a compact algorithm which generates maximum novelty. The reason for this is straightforward: when a compact algorithm generates strings, those strings are of a pattern with the other strings it generates.

Furthermore, if an algorithm learned from previous examples what is good and what is bad, and used this information to generate better objects, that would also defeat the end of producing "truly" novel objects. The very similarity which exists between known good objects and differentiates them from bad objects is a pattern which when identified can only be used to produce new good objects which are similar - in a precisely quantifiable sense - to the known ones. Maximally novel objects can in principle only be discovered using random search[4] or by already having a database of highly different objects and simply retrieving them from that database one by one.

[4]Using a physical random number generator. The pseudorandom number generators used in typical "stochastic" computer programs do not have algorithmically random output.

7.1 Perceptual Novelty

The impersonal "objective" version of novelty described in the previous section does not correspond to how novel an object will seem to an intelligent observer. Two different clips of random audio white noise sound the same to the human ear, even though in information theoretic terms they are likely to be maximally different from one another (there will be no common pattern to them). As a consequence, a successful theory of creativity will probably need to be a theory of creativity *relative to* some observer whose perceptual and conceptual capacities determine the effective novelty of creative products. We will see shortly that a formal impersonal version of novelty leads to direct contradictions which may be resolved by a perceptually-based theory. For instance, it has been proposed by Schmidhuber (2006) that perceptual novelty is related to the degree to which a new stimulus is expected to improve the observer's predictive model (as his paper observes, a successful predictive model must compress historical data).

Does this mean that human creativity must rely on non-algorithmic processes? Certainly not. What really matters is the perception of novelty by an observer, rather than the "objective" novelty of information theory. A short program can in principle produce a sequence of objects which appear highly dissimilar to a human perceiver, and a series of mutually random objects can appear highly similar. In other words, endless apparent novelty could be generated by a compact program by exploiting the limitations of the perceiver's ability to detect patterns. For instance, a human being zooming into the Mandelbrot set sees novelty for quite a while, because our visual apparatus is unable to pick up the simple algorithm which generates it.

8 Conclusion

The theoretical tools of algorithmic information theory are valuable to researchers in the field of computer creativity, not only because of their potential relevance to formalising aesthetics, but because they formalise the crucial concepts of *pattern* and *randomness*. These concepts are central to learning and computer generativity, and relevant to evaluating the novelty of new generative products. Compression deserves more prominence as an organising idea. For instance, this paper has argued that all generative algorithms can be seen as decompressors for a lossy compression scheme. However, under the most general information theoretic measure of novelty, concise computer programs (and presumably human beings) must always be understood as generating patterns which *appear* novel to a perceptually limited observer, rather than being *objectively* novel in some observer-independent sense.

Acknowledgements

I would like to thank Chris Thornton for helpful discussion and encouragement; my supervisor Inman Harvey, and an anonymous reviewer who pointed out some major omissions in my references. This research was conducted under a grant from the EPSRC.

References

Bennett, Gacs, Li, Vitanyi, and Zurek (1998). Information distance. *IEEETIT: IEEE Transactions on Information Theory*, 44.

Boden, M. (2003). *The Creative Mind; Myths and Mechanisms*. Routledge.

Cilibrasi, R. and Vitanyi, P. M. B. (2005). Clustering by compression. *Information Theory, IEEE Transactions on*, 51(4):1523–1545.

Li, M. and Vitanyi, P. M. B. (1997). *An Introduction to Kolmogorov Complexity and Its Applications*. Springer-Verlag, Berlin.

Saunders, R. (2001). *Curious Design Agents and Artificial Creativity*. PhD thesis, Faculty of Architecture, University of Sydney.

Schmidhuber, J. (1997). Low-complexity art. *Leonardo, Journal of the International Society for the Arts, Sciences, and Technology*, 30(2):97–103.

Schmidhuber, J. (2006). Developmental robotics, optimal artificial curiosity, creativity, music, and the fine arts. *Connection Science*, 18(2):173–187.

Svangard, N. and Nordin, P. (2004). Automated aesthetic selection of evolutionary art by distance based classification of genomes and phenomes using the universal similarity metric. In *Applications of Evolutionary Computing*, pages 447–456. Springer.

Witten, I. H., Bell, T. C., Moffat, A., Nevill-Manning, C. G., Smith, T. C., and Thimbleby, H. (1994). Semantic and generative models for lossy text compression. *The Computer Journal*, 37(2):83–87.

How Thinking Inside the Box can become Thinking Outside the Box

Chris Thornton
Department of Informatics,
University of Sussex,
Brighton,
BN1 9QH,
UK
c.thornton@sussex.ac.uk

Abstract

While it remains a central reference for work in computational creativity, Boden's exploration/transformation model can be interpreted as having different meanings at the application level. As a result, programmers developing creative systems may have difficulty in realising the theory's practical value. The paper develops a formalisation which recasts the key elements in quantitative terms while reinterpreting the exploration/transformation distinction as a continuum. This has the effect of making the framework more amenable to practical application in the system-building context.

1 Introduction

In Boden's original model (Boden, 1990) creativity is seen as taking two forms:[1]

- guided search in existing conceptual spaces (termed *exploration*) and
- creation of new conceptual spaces (termed *transformation*).

For Boden, the second type is the more important, being the origin of 'true originality' (Boden, 2003, p 40) and for many there is an echo here of the intuition that thinking 'outside the box' can be more creative than thinking 'inside the box'.[2] However, for those interested in the building of creative systems, the identification of these categories raises questions about what sort of guidance and transformation will be most advantageous. The practitioner may want to ask 'is my system starting out with the right conceptual spaces?' 'is search being guided in a productive way?' and 'is the system generating the right sort of transformations?'.

There may also be practical difficulties with the distinction between exploration and transformation. Since transformation can be seen as search performed in a space of conceptual spaces, it can be regarded as a form of exploration (Wiggins, 2001, Ritchie, 2006). Wiggins has in fact taken the step of proving the formal equivalence of transformation and exploration by showing that transformation can always be viewed as meta-search. (Wiggins, 2006b, Wiggins, 2006a) The system-builder may need some way of deciding when transformation should really be treated as exploration, or vice versa.

One strategy is to treat Boden's theory as strictly explanatory and not expect it to provide any particular processing model, or at least nothing more than the general notion that creativity involves guided search in conceptual spaces.[3] Another approach is to seek a formalisation of the theory capable of meeting the needs of the systems builder. This is the approach pursued here.

2 Concept duality

Boden's observation that conceptual spaces must be generatively represented (Boden, 1990, p. 78) implies that identifying a new concept in a conceptual space must involve *construction* of the concept. This process must presumably make use of existing concepts. So exploration of conceptual space must involve processes of concept construction in which new concepts are constructed from existing concepts. But what are the possible forms of this process? In what ways can concepts be combined to form a new concept?

Two cases can be discerned. First, there is the case where the sub-concepts are treated as instances. Second, there is the case where they are treated as constituents. The former type of construction is entailed in the construction of class-based, category-based and property-based concepts. The latter is entailed in construction of function-

[1]In Boden's revision of the model (Boden, 1998, 2003), a third form of creativity — combinational creativity — is identified. However, within the formalisation all exploratory creativity is combinational and for present purposes the two processes are therefore regarded as equivalent.

[2]Boden talks about the creative process mainly in terms of conceptual spaces explored by humans (and computers) but the model can be seen as covering search in *any* representational system. Ritchie, for example, interprets Boden's model in terms of search in a space of generic artifacts. (Ritchie, Forthcoming).

[3]This is the approach taken by Wiggins in his recent, information-based work on composition (Wiggins, 2007).

and relation-based concepts. For present purposes, construction in which the components are treated as instances will be termed **categorical** while construction in which the components are treated as constituents will be termed **compositional**.

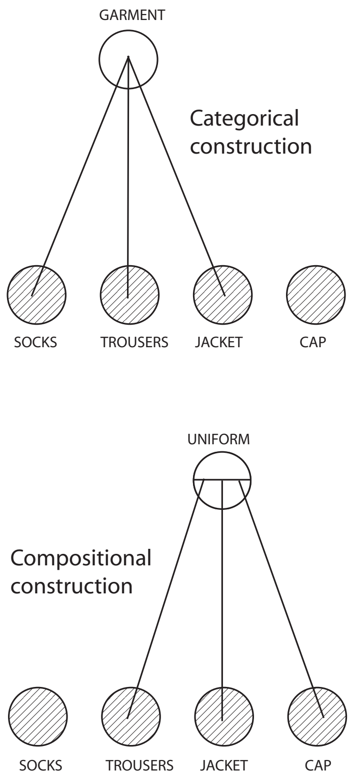


Figure 1: Concept-construction methods.

As an illustration of the distinction between categorical and compositional construction, consider Figure 1. This shows constructions for two clothing concepts: GARMENT and UNIFORM. In the diagram, concepts are represented as circles with primitive concepts being shaded. Three of these — SOCKS, TROUSERS and JACKET — may be combined to form the GARMENT concept. This is a categorical construction since the sub-concepts are treated as instances. In contrast, TROUSERS, JACKET and CAP may be combined to form the concept UNIFORM (as in 'MILITARY UNIFORM'). But here the construction is compositional since the components are treated as constituents of a new whole.

Note how the arcs connect to the internal structure of UNIFORM, reflecting compositionality while the arcs connecting to GARMENT combine, reflecting the fact that the concept is a class in which the components are alternative instances. (This convention is followed throughout.)

Whether a higher-order concept can be constructed in a particular way depends on the concepts and — in the case of compositional construction — the relations that can be applied. The fact that a particular concept can be constructed in one way from certain components does not mean that it cannot also be constructed in the other. Nor does it in any way limit the ways in which the components can be used. The UNIFORM concept is here shown in a compositional construction. But it could also have been shown in a categorical construction, using sub-concepts such as ARMY UNIFORM, POLICE UNIFORM and SCHOOL UNIFORM. And while the compositional construct makes use of the 'and' relation, applying this relation to different components, or using a different relation altogether, other constructs could be formed from the same primitives.

3 The general form of conceptual development

Being able to construct new concepts endows an agent with the ability to 'explore' a particular conceptual universe. But what can we say about this universe? How big is it? What is the structure? If the agent is solely capable of categorical construction, only a finite number of new concepts may be constructed and these must correspond to the possible subsets of primitive concepts. If the agent is capable of compositional construction, then there is the possibility of an infinite expansion of concepts. But the rate at which new concepts may be developed depends on the relationships that may be applied.

As an illustration of the possibilities, consider Figure 2. This represents conceptual development from three primitives (the shaded circles) using categorical construction, and compositional construction with two relationships (labelled 1 and 2). As before, categorical construction is indicated using arcs which combine. But here compositional construction is indicated using arcs which connect with a bar labelled with the relationship invoked.

Initially, there are just the three primitives. For every way of grouping these, there is the potential for a categorically constructed concept and two compositionally constructed concepts — one for each of the available two relations. These initial constructions generate concepts which are first-order with respect to the primitives. For every way of grouping the first-order concepts, the same situation applies, with the result being a layer of second-order concepts. The number of potential constructs thus grows multiplicatively, with each level containing concepts of higher order.

But what is the maximum rate of growth? Initially, there are just the primitive concepts themselves. Let k represent the number of these. Each directly derived concept must combine some of the primitives. So the number of derivable concepts must be related to the 2^k possible subsets, but discounting the empty set and all singleton sets.[4] That is to say, the number of subsets on which new concepts can be constructed must be

$$2^k - (k + 1)$$

Each subset provides the basis for one categorical construct and, for each available relationship, one compositional construct. Letting r represent the number of accessible relationships, the total number of concepts which can be directly constructed is thus

$$(2^k - (k + 1))(r + 1)$$

[4]Not discounting them would allow construction of empty and duplicate concepts.

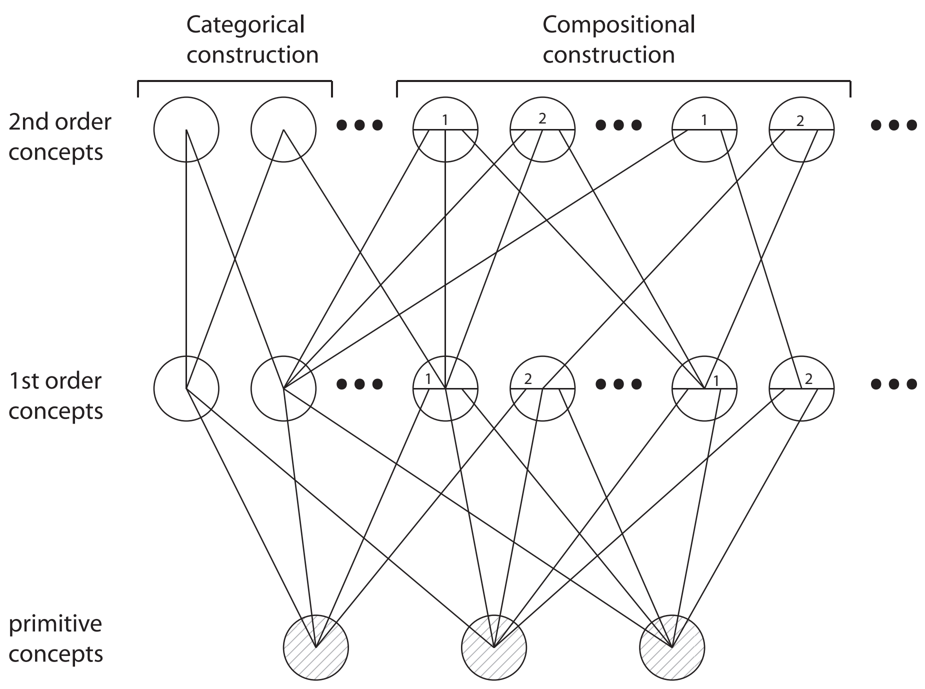


Figure 2: Potential concept constructions using 3 primitives and 2 relations.

Applying categorical and compositional construction to the first layer of derived concepts generates a second layer of concepts, and so on. The number of concepts which can be constructed at any level of the hierarchy thus depends on the number at the level below. This allows the rate of growth to be defined recursively. The number of concepts which can be constructed at any level i of the hierarchy may be determined using

$$\begin{aligned} k_0 &= \text{the number of primitive concepts} \\ k_{i+1} &= (2^{k_i} - (k_i + 1))(r + 1) \end{aligned}$$

Considering this growth formula, there can be no doubt that the number of possible constructions grows exponentially fast as development progresses. The number of constructions at a specific level of the hierarchy is multiplicatively related to the number at the level below both through the explicit exponentiation of k and through the multiplication with r. The result is that an unmanageably large number of potential constructions is reached very rapidly regardless of the initial base. For example, assuming a base of just three primitives and a single relationship, there are

- 8 potential concepts at level 1,
- 494 at level 2 and
- more than 10^{149} at level 3.

As a general rule, exhaustive expansion of a concept hierarchy beyond two levels of construction is intractable.

4 The Complex Extension

Examination of the mechanisms of concept construction reveals how conceptual spaces must be explored and answers some of the questions raised in the application of Boden's model. Exploration of conceptual space must proceed on the basis of categorical or compositional concept-construction. For any conceptual space, there must be an initial set of primitive concepts and, if compositional construction is used, a set of applicable relations too. In the case where only categorical construction is applied, the space is finite. If compositional construction is also available there is the potential for infinite development of the space. However, in this case, the rate of growth in any unrestricted process of construction is such that exploration of the space beyond the low-order concepts (i.e., 2nd or 3rd order) is prohibitively costly.

Understanding the general form of conceptual space exploration, however, does not answer the critical questions about *which* forms of exploration are likely to be most advantageous. The system-builder needs to know something about the general principles of heuristic guidance. This is the 'nuts-and-bolts' end of the evaluation problem, of course — the general problem of how to discriminate concepts which have genuine value.

While evaluation does not figure in Boden's core model (except in the sense that transformation is deemed to lead to more 'radically' creative conceptualisation) it does figure considerably in her commentary and illustrative examples. A notion which seems particularly significant there is that of *explanatory value*, i.e., the ability of one concept to account for, generalise or explain several others. This is to the good from the perspective of formalisation since generality has a well-defined meaning: the generality of a concept is *by definition*, the number of cases or instances which it generalises.

We can express the notion as

$$g(c) = |e(c)| \tag{1}$$

where $g(c)$ is the generality of concept c and $e(c)$ is its extension, i.e., its set of instances.

Can we incorporate into this formalisation a notion of explanatory value based on this equation? To do so we will need a mechanism for computing the extension of *any* concept. In the simple case of a categorical construct built directly on primitives, there is no difficulty. The extension is just the set of primitive concepts used in the construction. But how to compute the extension in the case where the concept is not defined directly in terms of primitives, or in the case where the construct is compositional?

In general, the instances of a concept are its possible manifestations and each distinct way of constructing a concept offers an alternative manifestation. Thus, alternative forms of construction *are* alternative forms of instantiation. Instantiation recapitulates construction. We can compute the extension of any concept, then, by evaluating the set of ways in which it can be constructed from the relevant primitives.

Illustrating the general idea, Figure 3 shows a concept hierarchy whose highest-level concept c is categorically constructed in terms of two compositional concepts which themselves are categorically constructed in terms of primitives. The extension of c contains its possible manifestations and these are identical to its possible constructions (as shown in the lower part of the figure).

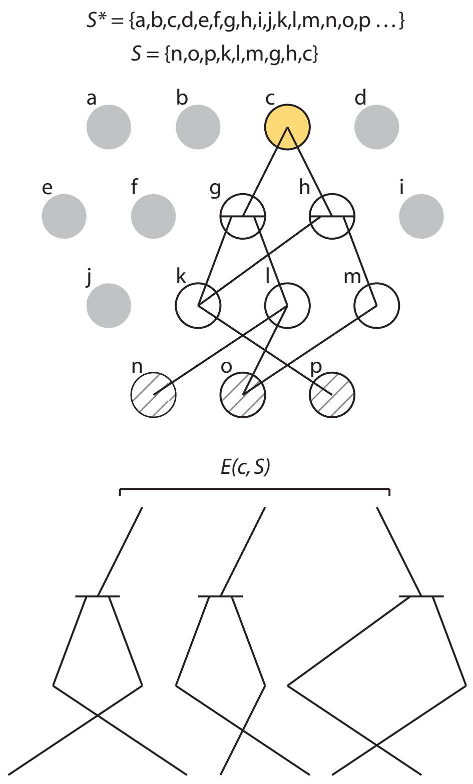


Figure 3: Derivation of extension.

There is a complication however since two forms of extension can be differentiated. On the one hand we have possible forms of construction made in terms of *existing* concepts of the space. On the other we have possible forms of construction made in terms of *potential* concepts of the space. In Figure 3 the set S is defined as containing all existing concepts of the space while $S*$ is defined as containing all potential concepts. Possible forms of construction for c in terms of already constructed concepts is therefore notated as

$$E(c, S)$$

with E being used instead of e to indicate the instances are conceptual structures rather than atomic entities.

In any conceptual space, the set of developed concepts must be a subset of the set of potential concepts. Thus

$$S \subset S*$$

and by the same token

$$E(c, S) \subset E(c, S*)$$

5 Full-house illustration

Where a single concept is constructed both categorically and compositionally, we have one extension for each construct. But though the contents must differ, the size of the extension (and by implication the generality of the concept) must be identical if all concepts of the space are realised. To understand why, consider Figure 4. This illustrates compositional and categorical construction of the FULL HOUSE poker concept, whose extension we know to contain exactly 156 cases (because there are exactly 156 instances of a five-card hand meeting the definition of 'full house').

The primitive concepts in this example represent specific cards, with 1d = 'ace of diamonds', 2h = 'two of hearts' etc. Concept construction invokes one relationship and this is simple conjunction (shown here as '&'). Possible categorical concepts at the first level are subsets of cards and these include the suit concepts HEART, SPADE, CLUB and DIAMOND. Compositional concepts are conjuncts of cards and these include all instances of pairs (e.g., 1d & 1c), three-of-a-kind, etc.

At the second level of development, there is the potential for categorically constructed PAIR and THREE OF A KIND concepts. There is also the potential for compositional constructs combining instances of both PAIR and THREE OF A KIND.

Finally, at the third level of development, there is the potential for both a categorical and a compositional FULL HOUSE construct, with the latter combining the second-level PAIR and THREE OF A KIND.

Although it cannot be determined from the diagram, it should be clear that the complex extension of this concept will contain the same number of cases whether it is calculated from the categorical construct or from the compositional construct. The situation with the categorical construct is straightforward. It utilises 156 components, each one of which corresponds to a unique full-house hand. The compositional construct, on the other hand, is built from just two categorical constructs but the possible constructions for these combine to produce the same overall total.

6 Hill-climbing in the generality landscape

Conceptual space development is subject to 'guidance' favouring explanatory value just in case it shows a pref-

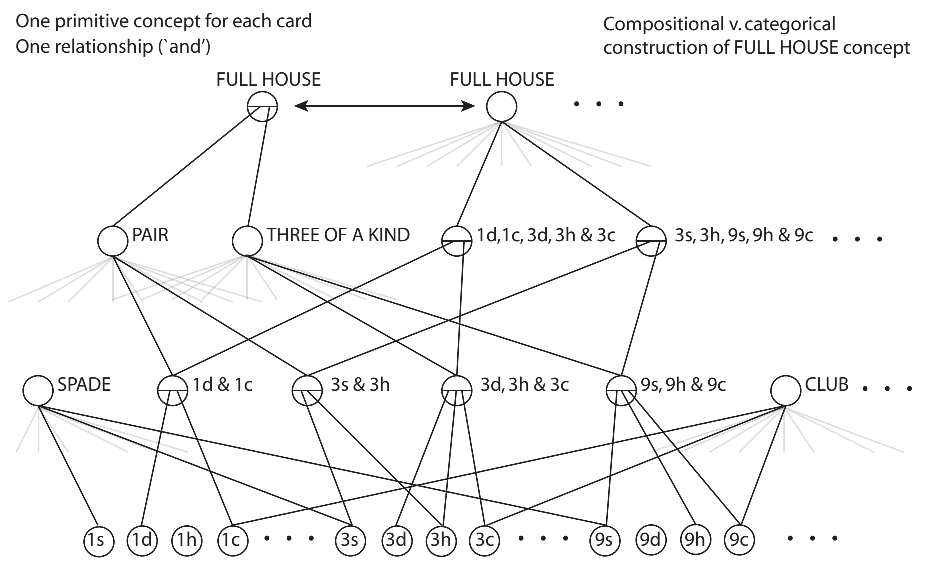


Figure 4: Dual developmental trajectories leading to FULL HOUSE.

erence for concepts of greater generality. If we envisage a conceptual space as a landscape of generality levels, we can take this to imply that the process is essentially a form of hill-climbing search. Figure 5 illustrates the point. Taking the more darkly filled circles to represent concepts of higher generality, explanation-oriented conceptualisation should construct concept c before a, b, or d because it has a higher level of generality. An operational definition of Boden's explanation-oriented exploration is thus that it is hill-climbing search carried out in a concept-generality landscape. But where does this leave the process of transformation? Where does the creation of new conceptual spaces fit in?

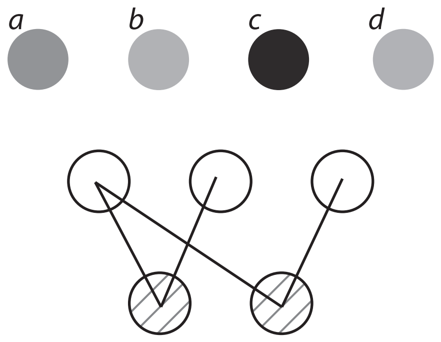


Figure 5: Explanation-oriented conceptualisation must hill-climb in the generality landscape.

Recall that generatively-represented conceptual spaces can only be explored through concept construction and that the components used in this process are existing concepts. On this basis any newly constructed conceptual space must exist within an overarching conceptual universe.[5] A conceptual space then forms a *subspace* in an enclosing conceptual universe. But taking into account Boden's point that the concepts in a conceptual space are connected through explanatory content, we can refine the definition and say that a conceptual space must comprise a subset of concepts whose extensional properties overlap.

In terms of the formalisation, then, transformation may be seen as *any* conceptual development which has the effect of creating a set of concepts with overlapping extensional content. On this basis, all conceptual development involves exploration but only some has transformational impact. The separateness and significance of transformation is thus upheld within the formalisation. But the link with exploration is more clearly delineated.

The general effect of this aspect of the formalisation is to re-cast Boden's all-or-nothing distinction between exploration and transformation as a continuum. But this seems not to disrupt the main content of the theory in any serious way. Indeed, some commentators have argued that an exploration/transformation continuum might be preferable (cf. Treisman, 1994, Weisberg, 1994), while others have made proposals that implicitly assume its existence (cf. Bundy, 1994, Koestler, 1964). There is also some cause for thinking that a continuum may be more compatible with psychological observations of creative activity (cf. Ram *et al.* 1995, Perkins, 1981)

7 Transformation distinctions

Viewing transformation as a type of exploration allows some sub-classifications of the process to be introduced. The transformation process is defined as being any development which has the effect of creating a set of concepts with overlapping extensional properties. But, in practice, this might mean two different things. It could mean the *construction* of the required concepts or it might just mean the *enhancement* of their apparent generality, since this will in any case have the affect of making construction more probable. There is a distinction to be made, then, between transformation which *actuates* a set of new con-

[5]Wiggins (2006b) draws the same conclusion on formal grounds.

cepts and transformation which merely helps to *potentiate* them by enhancing their apparent generality.

Further to this, there are two ways in which potentiating transformation may occur. On the one hand, there is the simple case where the constructed concepts *are* the potentiated concepts. On the other, there is the case where the potentiated concepts are in a different part of the hierarchy altogether. These two forms of the process are tentatively termed 'direct' and 'indirect'.

An an illustration of direct transformation, consider the example of Figure 6. This sketches the creative process underlying the innovation of the 'reality-TV' concept. This has been seen as a novel combination of media genres such as the soap opera, the game show and the human-interest documentary.[6] The diagram reflects this, showing the REALITY-TV concept as a compositional construct using the GAME SHOW, SOAP OPERA and HUMAN DOCUMENTARY concepts.

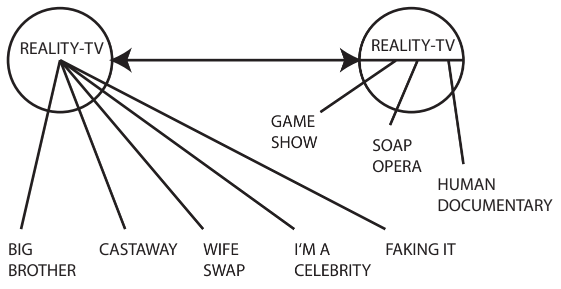


Figure 6: Illustration of direct transformation.

Prior to the compositional construction of REALITY-TV, concepts which might figure in its categorical construction (such as BIG BROTHER and CASTAWAY) are assumed to be unrealised and to have negligible generality. But once construction is complete, their generality is increased. All concepts within the category expand their extensions, producing an increase in generality. The compositional construction therefore generates *potentiating* transformation. Intuitively, the innovation of REALITY TV has value due to the fertility of the concept. In terms of the model, it has value due to its potentiating, transformational impact.

For a case of indirect transformation, see Figure 7. This features concepts relating to performances of the 'crossover' jazz pianist Jamie Cullum. Cullum is widely praised for his lively and imaginative improvisations which may involve use of pianos and other stage equipment (not to mention people) as percussive instruments. During intervals between passages of conventional keyboard wizardry, Cullum may engage in unconventional percussive activity, e.g., drumming on the music stand, banging the lid of the piano up and down and head-butting the microphone.

Figure 7 envisages Cullum's mould-breaking activities in terms of potentiating transformation. But unlike the previous case, the transformation here is indirect — the constructed and potentiated concepts being well separated. The compositional construct in the bottom-left

[6]The confrontation-prompting chat show, such as Jerry Springer, is another plausible constituent.

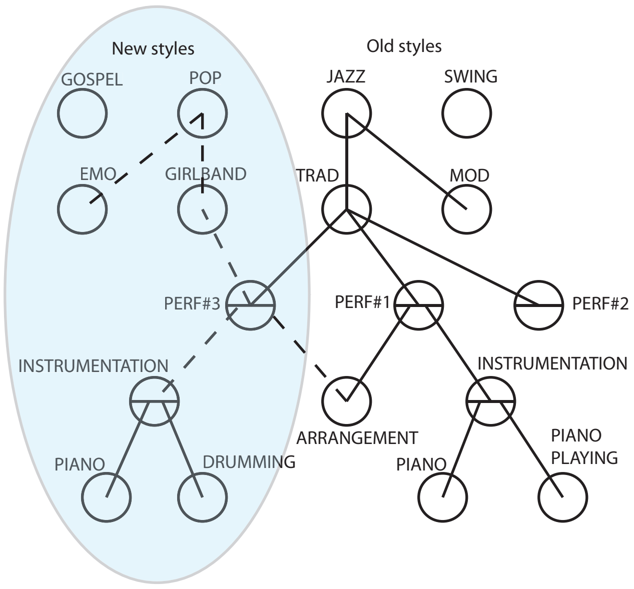


Figure 7: Illustration of indirect transformation.

corner represents the combination of PIANO and DRUMMING, i.e., it represents the idea of using a piano as a percussive instrument. Initially, this concept and all other concepts in the shaded area are unrealised. (The situation might correspond to an early stage in Cullum's career.) However, all concepts outside the shaded area are in existence and there is a well defined subspace of 4th and 5th-order concepts representing his early styles of performance (e.g. JAZZ and SWING).

With the innovation of the PIANO+DRUMMING composition, there is an immediate impact on the generality of concepts which trace their construction through this construct. Assuming DRUMMING has a variety of instantiations in terms of existing concepts, the generality of all these potential concepts is increased. The innovation of the PIANO+DRUMMING concept thus has a potentiating transformational effect, enhancing the generality of a set of new 4th nd 5th-order concepts (GOSPEL, EMO, POP etc.). In performance terms, the innovation opens up a whole space of performance variation involving combination of novel percussive activity with previous unvisited musical territory.

8 Summary

A formalisation of Boden's creativity model has been put forward which aims to answer the type of questions which can arise in applications work while at the same time staying true to the original account. Observations about basic mechanisms of concept construction were used to derive a mathematical model of conceptual development. This was extended so as to ground the notion of explanatory value and allow explanation-oriented creative conceptualisation to be understood as hill-climbing search. On the assumption of conceptual spaces being collections of concepts with overlapping extensional properties, transformation was interpreted to be any exploratory conceptualisation serving to create or accentuate such a set of concepts.

Several sub-classifications of the process were then distinguished. i.e., actuating versus potentiation transformation, direct versus indirect transformation.

An immediately apparent limitation of the formalisation is the restricted notion of value brought to bear. The only way in which creativity can be guided in the formalisation is through the generality measure. Therefore the only form of evaluation which is accommodated is that of explanatory power. While this may be appropriate in the context of scientific or intellectual creativity, it is presumably less relevant in other areas. Extending the formalisation so as to deal better with non-explanatory types of value is thus an important goal for future work.

A possible approach to the problem would be to look at the degree to which miscellaneous types of value can be effectively 'inherited' via the generality criterion. Recall that explanatory value is determined (for present purposes) in terms of extension size, i.e., its coverage of other concepts. And since these other concepts may themselves have *any* type of value, the valuation of the constructed concept measured purely in terms of coverage of subconcepts must reflect the types of value which those subconcepts have. So evaluation in terms of coverage might, under certain circumstances, implicitly support evaluation of a *non*-explanatory property.

For example, consider the cook who first discovered that sausages and onions are a good food combination. The artifact created is 'sausages & onions' and this presumably has elements of practical value (it satisfies hunger), associative value (it smells nice) and perhaps aesthetic value too (it looks nice). To the extent that these elements of value can be understood as originating in types of value associated with the *components* — sausages and onions treated independently — a generality-based model of evaluation might suffice.

However, while this approach may deal satisfactorily with inherited components of value, there is still the problem of accounting for that element of value attributable to the construction itself. Regarding this issue the most promising avenue may be to extend the formalisation so as to incorporate notions of conceptual blending (Fauconnier and Turner, 2005, Pereira and Cardoso, 2002) or combinatorial creativity (Butnariu and Veale, 2006, Veale and O'Donogue, 2000).

References

Boden, M. (1990). *The Creative Mind: Myths and Mechanisms*. London: Weidenfeld and Niicolson.

Boden, M. (1998). Creativity and artificial intelligence. *Artificial Intelligence*, *103* (pp. 347-356). 1-2.

Boden, M. (2003). *The Creative Mind: Myths and Mechanisms* (2nd edition). London: Weidenfeld and Niicolson.

Bundy, A. (1994). What is the difference between real creativity and mere novelty?. *Behavioral and Brain Sciences*, *17*, No. 3 (pp. 533-534).

Butnariu, C. and Veale, T. (2006). Lexical combinatorial creativity with gastronaut. *Proceedings of the Computational Creativity Workshop, ECAI 2006.*

Fauconnier, G. and Turner, M. (2005). *The Way We Think: Conceptual Blending and the Mind's Hidden Complexities*. Basic Books.

Koestler, A. (1964). *The Act of Creation*. London: Hutchinson.

Pereira, F. and Cardoso, A. (2002). Conceptual blending and the quest for the holy creative process. *Proceedings of the 2nd Workshop on Creative Systems: Approaches to Creativity in AI and Cognitive Science, ECAI 2002, Lyon Frane*.

Perkins, D. (1981). *The Mind's Best Work*. Cambridge, Mass.: Harvard University Press.

Ram, A., Wills, L., Domeshek, E., Neressian, N. and Kolodner, J. (1995). Understanding the creative mind: a review of margaret boden's 'creative mind'. *Artificial Intelligence*, No. 79 (pp. 111-128).

Ritchie, G. (2006). The transformational creativity hypothesis. *New Generation Computing*, *24* (pp. 241-266).

Ritchie, G. (Forthcoming). Some empirical criteria for attributing creativity to a computer. *Minds and Machines*.

Treisman, M. (1994). Creativity: myths? mechanisms?. *Behavioral and Brain Sciences*, *17*, No. 3 (p. 554).

Veale, T. and O'Donogue, D. (2000). In S. Coulson and T. Oakley (Eds.), *Cognitive Linguistics* (special issue on Conceptual Blending), *11* (pp. 3-4).

Weisberg, R. (1994). The creative mind versus the creative computer. *Behavioral and Brain Sciences*, *17*, No. 3 (pp. 555-557).

Wiggins, G. (2001). Towards a more precise characterisation of creativity in AI. *Proceedings of the ICCBR 2001 Workshop on Creative Systems*. Vancouver, British Columbia.

Wiggins, G. (2006a). A preliminary framework for description, analysis and comparison of creative systems. *Knowledge-Based Systems* (pp. xxx-xxx). Elsevier B.V.

Wiggins, G. (2006b). Searching for computational creativity. *New Generation Computing*, *24* (pp. 209-222). Ohmsha, Ltd and Springer.

Wiggins, G. (2007). *The Components of a Creative System: The Architecture of a Computer Composer*.

MINIMAL CREATIVITY, EVALUATION AND FRACTAL PATTERN DISCRIMINATION

Jon Bird
Creative Systems Lab
University of Sussex
Brighton, BN1 9QG, UK
jonba@sussex.ac.uk

Dustin Stokes
Centre for Research in Cognitive Science
University of Sussex
Brighton, BN1 9QG, UK
d.stokes@sussex.ac.uk

Abstract

Evaluation is considered to be an important component of any creative process. This paper explores how evaluation can be incorporated into our minimal model of creativity, which we have been developing using a combination of conceptual analysis and evolutionary robotics. Specifically, we consider how to extend our approach so that the robots themselves can evaluate mark patterns that they, or other robots, have made on the floor of their environment.

Evaluation can, we suggest, be distinguished into a descriptive and a merit assignment component. To evaluate an object or event F is, (a) to discriminate some feature of F and (b) to assign some merit or de-merit to that feature of F. Component (a) is descriptive and (b) is what would more traditionally be called 'evaluative.'

In simulation, our robots discriminate fractal from random patterns and demonstrate this by stopping on target regions of the arena floor that are covered with a fractal texture. We argue that in so doing they perform the descriptive component of evaluation. However, it is debatable whether the robots are performing the second, merit-assignment component of evaluation. Currently, the discrimination mechanism is hard-wired and does not develop during an agent's lifetime. In future experiments we will investigate how to artificially evolve agents that perform what might be described as 'minimal evaluation' by attempting to incorporate a preference element.

Keywords: Minimal creativity, evolutionary robotics, evaluation, fractal pattern discrimination

1 Introduction

A creative process, one might argue, involves evaluation. Without the second, you don't get the first. Consider Harold Cohen's AARON drawing system, which has now been generating images for more than 30 years (McCorduck, 1991). The computational structure of AARON has changed over the years. These details are theoretically interesting, but set them to one side for the moment. No matter the computation involved, and no matter the status of the artificial agency of AARON, the following is clear. Cohen, in selecting the images to display and eventually sell, must evaluate those images. He selects the ones that he prefers, and these images are the ones that you and I will see in a gallery. This is evaluation if anything is. And without it, one might say, you don't have a creative process. In fact, this may be the reason that one hesitates to call AARON creative: without Cohen, there is no evaluation. Indeed, Cohen makes the same concession (Cohen, 1999). This may motivate one to take evaluation to be a necessary condition for creative processes.

We are sympathetic to this general suggestion. And we are willing to take it seriously enough to incorporate it into our minimal approach to modelling creativity and to see what new results it may engender. Our ultimate goal is to evolve artificial agents which behave in, at least minimally, creative ways. Our methodology is also minimal, both in terms of conceptual assumptions, and the constraints we place on the robot controllers.

2 Minimal Modelling and Creativity

To this point, our conceptual assumptions about creativity have been sparse. First, we propose that creative processes require some degree of agency. And agency involves autonomy. We understand autonomy very broadly: it includes any behaviour not strictly imposed by a programmer or designer. This 'no strings attached' agency is weaker than a rich philosophical notion; it does not require deliberation or cognition. Thus a remote controlled robot would not be an agent in our sense, while an individual agent in an artificial life simulation would be. Second, we suggest that creative processes involve novelty. We follow Boden (2004) in rejecting the assumption that only historical novelty is theoretically interesting. Instead, one can acknowledge relative novelty by specifying different reference points. We are interested in two types of relative novelty. Some behaviour is *population-relative novel* for some agent A just in case that behaviour is novel relative to all of the agents in a population of which A is a member. And a behaviour is *individual-relative novel* for some agent A just in case it is novel relative to the behavioural history of A. This provides a great deal of flexibility in how we understand novelty. And of course, as the richness of the creativity in question goes up, so does the novelty, perhaps moving towards something like historical novelty. We thus propose an agency and a novelty condition for minimally creative processes (Bird and Stokes, 2006a,b; Bird et al., 2007).

To meet the 'no strings attached' agency condition our initial model consists of a simulated robot, situated in a walled arena environment, whose behaviour is solely determined by its sensory motor activity. The simulation is based on a Khepera robot with 6 IR sensors, a floor sensor that can detect lines directly below the robot and a pen that can be raised and lowered. Given the difficulty of hand designing robot-environment interactions we employ an evolutionary robotics approach to designing the artificial neural network that controls the behaviour of the robot. The fitness function explicitly rewards (a) correlated changes in the the state of the line sensor ('on to off' and 'off to on') and the pen ('up to down' and 'down to up'), and (b) marks made over a large area of the arena floor. It also implicitly penalises robots that crash into walls. This methodology also enables us to minimise any implicit or explicit biases we have about how creativity should be modelled as well as having the potential to generate models that exhibit unpredictable, and potentially novel behaviour. Our initial results demonstrate that we can evolve simulated robots that meet our minimal agency and population-relative novelty conditions (Bird et al., 2007).

3 Missing Evaluation

We've encountered many sceptics in presenting this work over the past year or two. The sceptic will often say something like the following. "You've got agents that perform reactive behaviours, some of them novel in your specified sense, but there is no space for these agents to make choices, to judge what they are doing, to evaluate. And this is what's missing: no evaluation, no creative process. So call the behaviours 'minimally creative' if you like, but they are too minimal to connect with rich creativity in any interesting way." As we said at the outset, we are sympathetic to this general line of criticism. It needs, however, to be separated into distinct criticisms, some of which we will address here, some of which we hope to address later in our research.

1. *Non-evaluative processing worry*
   This is the issue we are most concerned with at present. For agents to act creatively, they must be performing some kind of evaluation. The agent must be choosing among features or stimuli in its environment, where these choices inform (and indeed may be derived from) their behavioural activity. The mark-making behaviour of our agents lacks this feature. Their behaviours are mere reactions to the arena boundaries and previous mark-making in the arena, constrained by their own sensory motor morphologies and, from an evolutionary perspective, the performance of previous agents in that environment. There is, it seems, no evaluation by the individual robots of their mark-making activity.

2. *Myopic worry*
   A related worry concerns a feature of the sensory motor morphology of the agents we are using. As a symptom of their physical structure, our simulated mark-making agents can only see marks on the floor below them in a 2mm by 2mm area. There is thus no sense in which the agent can achieve a global perspective on the marks it is making. There is not, as there is usually for a human artist, an opportunity to 'stand back' and consider the overall pattern. This underwrites the non-evaluative processing worry, since this myopic perspective seriously limits prospects for evaluation.

3. *No-stopping worry*
   Our agents have no stopping mechanism. That is, there is no point where the agent will complete the marks, no analogue to the artist who happily steps back and says to herself 'It's finished!'. Our agents may stop, but any termination of mark-making is independent of the patterns made by the marks. This, in the theoretical context of creativity, is a problem.

4. *Aesthetic value worry*
   A final worry concerns the results of the agents' behaviours rather than the behaviours themselves. Some members of our research team would ultimately like to see some robot drawings which can be exhibited in a gallery. Controversies about aesthetic value and definitions of art to one side, displaying works in a gallery generally requires that the works possess some aesthetic merit by one criterion or other. Our results may be interesting, provided an observer has enough information about the artificial agency that generated them. But on their own, it is debatable whether they have any aesthetic interest. The challenge, then, is to achieve some aesthetically interesting results, while maintaining our theoretical stance towards creativity. In other words, we want our robots to create something aesthetically valuable, while minimising our influence over how they do it. So far, we fall short of this goal.

4 A Proposed Solution: The Fractal Framework

We think one solution to these various worries lies in the use of fractals. In simple terms, we intend to endow our agents with a capacity for discriminating fractal patterns, and a preference of sorts for completing such patterns. This is a broad enough constraint on our agents to allow, as it were, creative freedom. Fractal patterns – understood broadly as patterns which display self-similarity across a range of scales – can vary greatly in their appearance (for example, texture), and so our agents will only have a general pattern 'preference'. The resulting images thus stand to be surprising or unexpected, but the behaviours of the agents are nonetheless constrained: some images are fractal, others are not [1].

Before discussing the details of our modelling strategies to these ends, we should ask, from a purely conceptual point of view, how a fractal framework helps with the four evaluation worries described in Section 3. We consider each in turn.

1. *R1: The non-evaluative processing worry*
One simple reason our agents do not evaluate is that they don't have anything to *look for*. They are not, at present, pattern discriminators and so *a fortiori* are not pattern evaluators. Incorporating fractal pattern discrimination and, eventually, a preference for making fractal mark patterns dissolves this particular aspect of the problem. The agents are thereby endowed with a minimal evaluation technique, preferring certain marks over others and acting in ways that display that basic choice [2].

2. *R2: The myopic worry*
In order that our simulated agents can discriminate fractal patterns, rather than just detect marks, we are replacing the 2mm x 2mm mark detector with a 50mm x 50mm camera that points at the floor. However, even with the addition of this camera, the agent's viewpoint is still extremely limited and does not counter the myopic worry. This therefore remains a challenging problem and may have to be resolved by either using a bird's eye view camera or some type of memory in the agent's controller so that it can evaluate a larger part of the arena at any one time. This issue is independent of the fractal framework that is the focus of this paper and we do not address it further here.

3. *R3: The no-stopping worry*
Incorporating fractal patterns provides a natural stopping point. An agent making a fractal pattern will stop once it has generated a pattern that is self-similar across that range of scales that it can discriminate. The agent thus finishes the mark-making *and* in a way that is dependent upon the patterns made.

4. *R4: Aesthetic value worry*
This particular worry is perhaps too far downstream from our current state of research, so a suggestion for now will suffice. People like fractals. This seems true on an intuitive level and has in fact been demonstrated in the psychological literature (Spehar et al., 2003). What's more, if an audience member is armed with the knowledge that the agents she is watching are attempting to complete a fractal pattern, a very basic identification is achieved [3]. That is, the agents' behaviours are no longer unintelligibly random, since one understands an agent's general goal and may watch as the agent attempts to achieve that goal. A fractal framework thus affords potentially aesthetically valuable finished products, as well as a sharpened insight into the behaviours – and, perhaps, the creative processes – of the artificial agents. This, we suspect, moves us closer to the aesthetic merit needed for a gallery display.

5 Theoretical Analysis of Evaluation and Pattern Discrimination

What do we mean by 'evaluation'? Evaluation is a topic of rich debate in the philosophical literature ranging from aesthetics to moral theory to action theory, among others. This diversity provides a hint: evaluation is a context-bound enterprise and so theorizing it should be shaped by the relevant context. We are primarily concerned with art and art-making, so our analysis will err towards an analysis of aesthetic evaluation.

Most intuitively, to aesthetically evaluate some thing is to assign value to that thing. Less circularly, to aesthetically evaluate some thing is to assess the merit of that thing. We say that this work is 'good', that one 'bad', this one 'beautiful', that one 'poor.' We can, and often do, offer such assessments in general and unqualified ways. However, these assessments are typically made for particular reasons; and we provide these reasons if our assessment is called into question. I assign merit to some work because it has this or that property, and I appeal to the latter in justifying the former. This reveals something important about aesthetic evaluation: aesthetic evaluation, at least typically, involves a descriptive element.

There is an historical dichotomy between the evaluative and the descriptive. The distinction is – as is often the case with supposed dichotomies – a fuzzy one. The 20th century aesthetician Frank Sibley offers an insightful analysis of this distinction, with special emphasis on its place in philosophical aesthetics. Sibley's emphasis is on evaluative versus descriptive terms, but so long as we take a use of such terms to be indicative of the corresponding judgment, we can generalize from his analysis to evaluative and descriptive acts (Sibley, 2001).

First, Sibley suggests, some terms are used to indicate that an F has value (or does not), without indicating why or how F has this value. One may, for example, call a thing 'good', 'bad', 'nice', 'nasty', 'worthless' and so on, without attributing any particular properties to that thing. Sibley calls such terms solely evaluative. These terms and their correlative use are better understood when contrasted with a second class of terms. Some terms indicate a property the possession of which is a merit (or de-merit) relative to some category. 'Sharp' is such a term relative to the category of knives (and, oppositely, so is 'dull'), 'level' for billiard tables, 'round' for basketballs, and so on. Such terms, Sibley suggests, are often taken for evaluative ones. However, we might instead think of them as straightforward property terms, since to use them is to as-

[1]This, in fact, is a point of controversy in the literature on fractals (Halley et al., 2004). Given the methods of fractal measurement we employ, however, some patterns will meet the standard while others will not. Whether this satisfies parties on both sides of the debate over what we might call 'fractal realism' is an open question.

[2]Whether this amounts to anything we reasonably call 'evaluation' is debatable. The non-evaluative processing worry is central to this paper, so more on this point in sections 5, 6, and 7.

[3]If and when we should be in a position to exhibit in a gallery, the display would be not merely of resulting images, but of the embodied robots making marks on the floor of a walled arena.

cribe a property to an object and indeed can be done without an additional evaluative assessment of that object. One can describe a knife as sharp without knowing that that sharpness enables proper performance of a knife's function. In Sibley's words, "[i]n general, one does not need to know, with such a term, 'P', though one often will, that the property counts as a merit in something in order to be able to ascertain that the thing may correctly be called 'P'" (Sibley, 2001, p.92). Indeed use of these terms is common in many spheres, including aesthetics and criticism. Consider critical and appreciative practice: we do offer descriptions of artworks (for example, of their formal, art-historical, generative, or socio-political properties) which may imply an assessment of merit, but which could be offered without such an assessment. The important point of contrast is that to use a term of this type, unlike solely evaluative terms, is to identify a particular feature of that object, and this is a descriptive rather than an evaluative act.

As Sibley admits, use of a descriptive merit term often, even if not by conceptual necessity, implies an assignment of merit. To this end, we might, again following Sibley, introduce a third category of terms: evaluation-added terms. Such terms involve both a descriptive and an evaluative element. In using a term of this type, one describes an F by indicating that F has some property G, and then adds to this description an assessment of merit, where this assessment takes place on the basis of the description of F's possession of G. Sibley offers 'tasty' as one example. If I call a meal 'tasty', I am no doubt giving a positive evaluation of the meal and this evaluation implies a certain description, namely, that the meal has a lot of flavour.

Sibley goes on to question whether aesthetic evaluation carves up so neatly, and indeed questions how much of so-called aesthetic evaluation is even evaluative rather than descriptive. No matter. We can safely glean the following lesson from Sibley's analysis. Aesthetic evaluation generally involves both a descriptive and a merit-assigning element; it roughly resembles the use of what Sibley calls 'evaluation-added terms.' Consider a familiar scene from a gallery or exhibit. My friend Jon says to me, "This sculpture is lovely." An eyebrow raised, I respond, "Really, how so?" Jon's response might go something as follows. "Well, it possesses a certain balance. Notice how the curve of her hip echoes the positioning of her opposite elbow. And the face is just expressive enough: the eyes are blank but wide open; the lips are slightly curled at the corners, not quite smiling and not quite frowning." Jon may go on and on until I have had enough. "You're right. It's lovely. Time for a drink". This is a simple example of appreciative practice. One may initially offer an unqualified evaluation of a work (using what Sibley calls a 'solely evaluative term'). However, as is often the case, one has reasons for making that evaluation and will offer them when asked. These reasons, as with Jon's reasons, often, perhaps always, involve descriptions of the thing evaluated. They involve an indication of the properties which underwrite one's assessment of merit. This is enough to motivate the following understanding of evaluation.

Evaluating an F involves:

1. d indicating that F possesses property G; and
2. m indicating that one finds merit/de-merit in G as possessed by F.

d is thus the descriptive element, and m the (traditionally) evaluative element. One can think about d and m as necessary and conjointly sufficient conditions for evaluation if one prefers, but we see no need to make this commitment. For our purposes, it suffices to say that evaluation, especially in contexts of aesthetic appreciation and criticism, typically involves both a property description d and an assignment of merit m. Indeed, this seems to capture a fundamental schema for evaluation, in whatever realm it should be. The difference, for example, between moral evaluation and aesthetic evaluation lies in the kind of merit that is assigned and, perhaps, in which properties are relevantly discriminated. Common to both kinds is the presence of a descriptive and an evaluative element.

We are taking a property description to be no more than the indication that an F has property G or, if one prefers, the discrimination of G as possessed by F. This kind of discrimination is clearly descriptive (as contrasted with evaluative), but it need not be any kind of rich description. Indeed it need not be linguistic: a picture, pointing at something (given the right context), the firing of a feature detecting neuron, or pushing one button rather than another could each just as well serve this indicative role. One may have reservations about calling these acts and events 'descriptions' given the heavy philosophical baggage that comes with this term. We are sensitive to these worries, and indeed our analysis of evaluation does not require us to think of descriptions in any special way. To be clear: the only point that need be granted is that evaluation involves (partly) an indication that the object under evaluation possesses some property or properties. And this activity, following Sibley, can be performed in purely descriptive, non-evaluative ways.

This kind of evaluation is no less a feature of art-making than it is of art appreciating. When making an artwork, an artist constantly makes choices which are informed by evaluation of the work up to its present state. These evaluations involve an assessment of merit – where this assessment is informed by property descriptions – of the properties possessed by the work in progress. There may be a difference in the degree to which or frequency with which an artist justifies her ongoing evaluations by appeal to the underwriting descriptions, but this is merely a contingent social fact. If we forced artists to work under the sociological microscope they would, like Jon, offer descriptions of the work in progress which justified their assignment of merit or de-merit and the corresponding decision that came with that assignment. The fact that they are more often pressed, *post facto*, to explain their evaluations does not imply that they made them in any other way.

6 Fractal Pattern Discrimination

In this section we describe our approach to implementing real-time fractal pattern discrimination on a simulated robot. The key property of a fractal object is that it is self-similar over a range of spatial scales. Three types of self-similarity are found in fractals. An object can be *exactly* self-similar at different scales, for example, Cantor dust (Figure 1), the Sierpinski carpet, Koch snowflake and other fractals which are generated by an iterated function system (which uses a geometric replacement rule). Objects can also display *approximate* or *quasi-*, rather than exact, self-similarity at different scales. These fractals contain distorted copies of the entire fractal at different scales. For example, fractals generated using an escape-time technique, such as the Mandelbrot and Julia sets, are quasi-self-similar. In the weakest form of self-similarity, statistical measures (such as 'fractal dimension') are preserved across scales. For example, fractals generated by processes such as diffusion-limited aggregation are *statistically* self-similar.

Only mathematical fractals display self-similarity across an infinite number of scales. Natural fractal objects display quasi- or statistical-self-similarity over a limited range of scales.

In contrast to Euclidean objects, fractals usually have non-integer dimensions. The fractal dimension measures the extent to which an object fills the Euclidean space in which it is embedded (Mandelbrot, 1982). A set of points along a line will have a fractal dimension between 0 and 1; a set of points on a plane have a fractal dimension between 1 and 2.

6.1 The Box-Counting Approach to Measuring Fractal Dimension

Box-counting is the simplest and most widely used technique for measuring fractal dimension and involves superimposing a series of regular grids over the data set. A regular grid consists of square boxes with a side length s. The measurement process is carried out using grids with a range of different side lengths. The first grid is layed over the set of data points and the number of occupied boxes, $N(s)$, counted. A box is occupied if it contains at least one data point. $N(s)$ is then plotted against $1/s$ for all box sizes. On a log-log graph, the slope of the graph is an estimate of fractal dimension. A fast $O(n \log n)$ algorithm (where n is the number of data points) was proposed by Liebovitch and Toth (1989) and implemented in C by Sarraille and DiFalco (Sarraille and Myers, 1994). This FD3 code is open source and available from: ftp://www.cs.csustan.edu/pub/fd3/. We have adapted FD3 to enable our robots to perform real time fractal pattern discrimination.

In order to confirm that a structure is fractal, it is necessary to show that it is self-similar over a reasonable number of scales. What constitutes a 'reasonable number' is a matter of some controversy. The range of scales is defined as: $log_{10}(Lmax/Lmin)$, where $Lmax$ is the largest or coarsest scale, and $Lmin$ the smallest or finest. In the physical sciences, the scale ranges tend to be small (Mandelbrot, 1998) and this can lead to incorrect estimations of fractal dimension or erroneously describing non-fractal structures as fractal ('apparent fractality') (Hamburger et al., 1996). Halley et al. (2004) recommend a scale range of greater than two orders of magnitude to avoid these problems, but this is not always possible.

It is important to note that although the box counting technique, described above, can employ a very large range of box sizes, the usable range is generally a lot smaller. For example, the largest box size used in FD3 is 2^{32} larger than the smallest box size (giving over 9 orders of magnitude scale range) but estimates from the smallest and the largest boxes have to be discarded, often resulting in a usable scale range of less than one order of magnitude. At very fine scales, none of the boxes contain more than one data point (depletion); at coarser scales all of the data points can be contained in one box (saturation). At these limits, the box counting algorithm will incorrectly estimate the fractal dimension. Consequently, fractal analysis is generally limited to a range of box sizes. The two largest box sizes are ignored. The smallest box size s used to estimate the fractal dimension meets the condition $N_B(s) \ll N/5$, where N is the number of data points and $N_B(s)$ the minimal number of boxes required to cover the data set at scale s (Liebovitch and Toth, 1989). FD3, which we use to measure fractal dimension, follows this convention. Our robots process small pixel arrays where the usable scale range is typically between 1.2 and 1.8.

Hamburger et al. (1996) investigated the fractal dimension of a number of small discs randomly scattered on the plane and demonstrated that this intrinsically *non* self-similar pattern can exhibit an almost linear relationship between $1/s$ and $N(s)$ over two orders of magnitude [4]. "Whether an apparent straight line on logarithmic axes really suggests a fractal or not is obviously a difficult and fundamental question" (Halley et al., 2004, p.259). Knowledge of the process that generated a pattern can sometimes help determine whether it is legitimate to describe it as fractal or not. However, it is important to note that using a fractal dimension measurement tool, such as Fd3, 'off the shelf' and without any consideration of the pattern that is being measured can lead to erroneously labelling a non-fractal object as fractal.

6.2 Lacunarity Analysis

A further issue, relevant to our project, is that two genuinely fractal objects can have the same fractal dimension and yet be very different in appearance, for example, Cantor dusts (Figure 1). One way that such patterns can be discriminated is in terms of their texture. Lacunarity is a useful measure of texture, introduced by Mandelbrot (1982), that quantifies the heterogeneity of the gaps in a pattern. Patterns that have gap sizes that are distributed over a greater range have a higher lacunarity index than patterns where the gap sizes are more similar. Objects that have a low lacunarity index are translationally-invariant because of their uniform gap sizes (Plotnick et al., 1993). Intuitively, one could shift sections of a pattern without altering its overall appearance. For a high lacunarity pattern,

[4]It is a matter of debate in the fractal literature whether patterns generated by random processes, for example, Brownian motion and self-avoiding random walks, should be considered apparent or actual fractals. We are clear that we want our robots to generate non-random self-similar mark patterns and we therefore want them to discriminate these patterns from those generated by random processes.

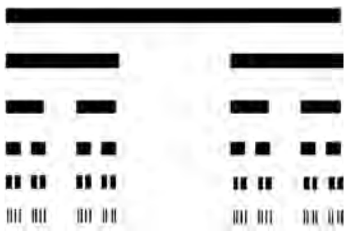


Figure 1: Cantor dust - an exactly self-similar fractal. Each of the lines has the same fractal dimension (0.6309) but a different texture, which is dependent on how many times the replacement rule (remove the central third of each line segment) has been applied. For illustrative purposes, the lines have been thickened.

shifting sections would become very apparent because the pattern is *not* translationally-invariant. It is important to note that lacunarity is a scale-dependent index - an object can have a homogenous texture at one scale but a heterogeneous texture at another scale. Further, one can measure the lacunarity of objects which are not self-similar.

There are a number of different algorithms for calculating the lacunarity of a pattern and there is not always agreement between the values that they generate (Halley et al., 2004). The most widely used technique is the gliding box algorithm (Allain and Cloitre, 1991). In this method a series of square boxes with varying side lengths s are placed over the data set and the number of points in each box (the box mass) is counted. The first box is placed in the top left hand corner and the box mass measured and then the box is systematically moved over the data set so that the position varies by one column or row. Unlike in the box counting method for measuring fractal dimension, in the gliding box algorithm the boxes overlap. For a square pattern with side M, there are $(M - s + 1)^2$ positions for a box of side length s where the box mass is measured. For each box size s, the mean and variance of the box mass is calculated. Lacunarity (Λ) is calculated as:

$$\Lambda = variance(s)/mean(s)^2 + 1.$$

When s is equal to 1, that is, a single pixel, or the grain of the data set, Λ is a function of the number of points in the data set and is independent of their spatial distribution. In this case, $\Lambda = 1/\%\ of\ on\ pixels$, where, in the case of a black and white image, an $on\ pixel$ is black. When a box is the size of the data set, the variance is 0 and so Λ is 1. In between the lower and upper bounds, lacunarity varies according to the range of gap sizes (or alternatively, clump sizes) at a given scale.

If an object is a *homogenous* fractal then the same scaling law applies at all positions of the object. Further, a log/log plot of lacunarity versus box side length generates a straight line where the slope is equal to the fractal dimension (D) minus the Euclidean embedding dimension (E); that is, D - 2 for all patterns on the plane (Allain and Cloitre, 1991). We use this relationship between fractal dimension and lacunarity to hard-wire a fractal discrimination mechanism in our robots.

6.3 Discriminating Random from Non-random Spatial Patterns

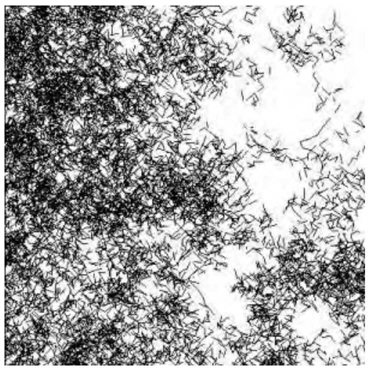


Figure 2: A randomly generated pattern of line segments which has the same % of black pixels (36%) as the two images in Figure 3. This pattern is discriminated from a self-similar fractal pattern, such as the left hand side image in Figure 3 using the method outlined in Section 6.3.

In this section we describe how we are using a box counting approach to get real-time estimates of fractal dimension and lacunarity of the spatial pattern in a simulated robot's visual field. Given the small number of points in the data set (limited by the small visual field of the robot) and the generally limited scale range, the technique we are employing could erroneously estimate self-similarity. At this preliminary stage, our goal is to develop a real-time method that enables our agents to discriminate random from non-random self-similar mark patterns. The class of non-random self-similar patterns that the robot can discriminate should have a structure such that:

1. the robot can produce members of this class with its pen;
2. and be sufficiently 'interesting' such that some members of this class are suitable for display (see 'aesthetic value worry' in Section 3).

At each sensory-motor update, the robot controller processes its 50mm x 50mm visual array in the following way:

1. using a box counting algorithm it estimates the fractal dimension (D) of the pixels over a range of scales where the ratio of the largest to the smallest box length side (s) is 2^{32}: 1 (the data points are re-scaled in order to achieve this range of box sizes);
2. using a box counting algorithm it estimates the lacunarity (Λ) at the same range of scales [5];
3. using regression analysis it measures the goodness of fit ($R2$) of the lacunarity curve over the range of scales which form the basis of the fractal dimension estimate;

[5]By using a box counting approach we estimate Λ on the basis of far fewer samples than is used in a gliding box algorithm. However, real time processing constraints forced this compromise in these preliminary experiments.

4. if $R2$ is greater than 0.95, it calculates the % error between the slope of the lacunarity curve and the estimate (D - E) of the lacunarity curve;

5. if $R2 > 0.95$ AND % error $< 20\%$, then the fractal pre-processing of the pixel array returns 1, otherwise 0.

This processing technique successfully discriminates a wide range of self-similar images from randomly generated images – not only images consisting of randomly distributed points but also randomly distributed lines segments, such as Figure 2, which are more likely to be generated by our robots than points.

6.4 The Fractal Discrimination Task

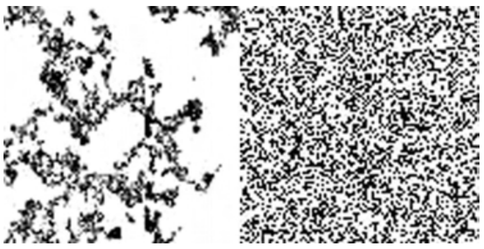


Figure 3: The left image is the texture on the target floor region in the evolutionary robotics experiment; the right image is the random point texture on the rest of the arena floor. Both patterns have the same percentage of black pixels (36%). The method outlined in Section 6.3 enables a simulated robot to discriminate between these two patterns.

The experiment described in this section was performed in a modified version of the Evorobot simulator (Nolfi, 2000). This software simulates a Khepera robot (Mondada et al., 1993) acting in user specified environments that can comprise of walls, large and small round objects and lights. Sensor readings taken from a physical Khepera robot are used to model the environment/sensor interactions.

The robots are controlled with neural networks based on Nolfi's (1997) emergent modularity architecture which has been successfully used to control complex robot behaviours, such as garbage collection. In the experiment reported here, each controller consists of 7 sensors (6 IR and 1 floor camera that is directly under the centre of the robot and has a visual field of 50mm x 50mm) and two pairs of motor units, controlling the right and left motor respectively. Each sensor connects to each motor neuron, giving 28 connections in the network. For details of the neural network update algorithm see Bird et al. (2007).

We used a genetic algorithm (GA) to evolve the biases of the 4 motor units and the connection weights between the sensor and motor neurons. The population size was 100 and the experiments were run for 600 generations. The initial population was randomly generated, each genotype consisting of the 32 neural network parameters encoded as an 8 bit integer-valued vector (range [0,255]). The mutation rate was 0.01 per allele and we did not use crossover. For more details see Bird et al. (2007) where the same GA was used to evolve robot mark-making behaviour.

The fitness function rewards proximity to the target area in the arena, with extra fitness if a robot is positioned on the target area at the end of a trial. The target area is 100mm x 100m and placed in one of two positions adjacent to the wall of a 400mm x 400mm arena. The target region consists of a fractal texture (Figure 3- left image); the rest of the floor has a random point pattern with the same overall ratio of black to white pixels as the target texture (36%)(Figure 3 - right image). If the robot crashes into an arena wall the trial is stopped and the fitness accumulated up to that point is averaged over the total number of time steps, thereby implicitly penalising robots that do not avoid obstacles.

Each genotype is instantiated as a robot controller and the robot is placed in a random position and orientation in the central area of the arena. Each individual is tested over 10 trials and the position of the target region is placed in two different positions, both adjacent to the wall of the arena. Every robot in the population was tested on the same series of initial positions and orientations each generation, and these changed every generation.

6.5 Preliminary Results

In early generations the robots do not move very far from their initial position or if they do they crash into walls. However, within 100 generations the majority of the population avoid obstacles and perform wall following. After 500 generations, the fittest individuals move in a straight line until they come close to a wall then follow the wall until they are over the target area and then stop. This is not a particularly surprising result as the patterns covering the target region and floor were chosen so that the fractal discrimination mechanism could clearly discriminate between them. However, it does demonstrate that this mechanism can be used to control the real-time behaviour of a robot and enable it to discriminate between random and fractal patterns on the floor. As in our previous experiments (Bird and Stokes, 2006b) the robots have evolved to use the arena walls (a constant and reliable feature of the environment): in the current experiment they provide a means of finding the target region which is always positioned adjacent to the edge of the arena.

7 Discussion

What does the preliminary theoretical analysis of evaluation tell us about our artificial agents and their pattern discrimination behaviour? By discriminating a fractal pattern from a random pattern are our agents evaluating their environments? Not obviously. But they are on their way. By discriminating fractal patterns, our agents are providing property descriptions, sparse though such descriptions may be, of their environment. When an agent stops on a fractal pattern and not on random patterns, it indicates the presence of a property – namely homogenous self-similarity – in the target location. This is just to say that this behaviour 'reports' that some part of the environment is a certain way, and other parts are not. The report is no richer than this, but neither is my report that this object is a square, while that one is not. Both kinds of reports are minimally descriptive of the world.

As we outlined in Section 5, evaluation comprises a descriptive element. Our agents are thus performing one element of evaluation: their pattern discrimination is a simple property description of their environment. However, what about the second, assessment of merit, component of evaluation: do our agents do this? This is a trickier issue. But here is a speculative suggestion. If our conception of evaluation is accurate, then it requires assigning merit to the properties of the object that have been discriminated. One assigns merit to the things that one likes or prefers, and de-merit to the things that one does not like or prefer. Preferences drive an agent, motivating it to act in whatever ways it does. Assignment of merit is thus to indicate, at base, what philosophers and psychologists call a 'conative attitude' [6]. This is true of human beings as well as laboratory rats; evaluation, no matter how rich, involves conation. Our agents, to evaluate in any interesting sense, thus need a preference or some degree of conation.

It would be misleading to describe the agents in our simulation experiments as possessing *individual* preferences. That is, our agents have not developed in their 'life spans' a preference or conative attitude of any kind. However, in this respect, do the agents differ from a laboratory rat, whose conative attitude of hunger motivates it to push the lever on the right and not the one on the left? The rat's preference is no more individually developed than is the fractal preference of our simulated agents. In the case of both the rat and the artificial agent, the preference has, in some sense, evolved in the population to which each belongs[7].

We might therefore say that fit robots, by successfully distinguishing fractals from non-fractals (and later in our research, by completing fractal patterns by mark-making) are thereby performing a number of simple evaluations which inform their respective behaviours. This is not rich evaluative behaviour, but if the rat is evaluating, then so is our robot. This, we suggest, is the basic route from mark-making and detection via pattern discrimination to pattern evaluation.

Acknowledgements

The *Computational Intelligence, Creativity and Cognition* project is funded by the AHRC and led by Paul Brown in collaboration with Phil Husbands, Margaret Boden and Charlie Gere.

[6]Conative attitudes are motivational states and are traditionally contrasted with both cognitive, or knowledge acquiring, states and states of affect. They comprise desires, values, preferences, likings, and so on. They are proactive and without them, agents do not act.

[7]In the case of the rat both the preference and the mechanism that underpins it evolved whereas in our robots the discrimination mechanism was hard-wired by us and the robot evolved the ability to use it appropriately. By prescribing what patterns it can discriminate, it might be argued that we have compromised the autonomy of the agent and consequently future fractal mark-making behaviour cannot be described as 'creative' (see Section 2), even though the fractal pattern discrimination might be minimally evaluative.

References

Allain, C. and Cloitre, M. (1991). Characterising the lacunarity of random and deterministic fractal sets. *Physical Review A*, 44:3552–3558.

Bird, J. and Stokes, D. (2006a). Evolving fractal drawings. In Soddu, C., editor, *Proceedings of the 9th Generative Art Conference*, pages 317–327.

Bird, J. and Stokes, D. (2006b). Evolving minimally creative robots. In Colton, S. and Pease, A., editors, *Proceedings of the Third Joint Workshop on Computational Creativity*, pages 1–5.

Bird, J., Stokes, D., Husbands, P., Brown, P., and Bigge, B. (2007). Towards autonomous artworks. *Leonardo Electronic Almanac*. In press.

Boden, M. A. (2004). *The Creative Mind*. Routledge, London, second edition.

Cohen, H. (1999). Colouring without seeing: A problem in machine creativity. *AISB Quarterly*, 102:26–35.

Halley, J. M., Hartley, S., Kallimanis, A. S., Kunin, W. E., Lennon, J. J., and Sgardelis, S. P. (2004). Uses and abuses of fractal methodology in ecology. *Ecology Letters*, 7:254–271.

Hamburger, D., Biham, O., and Avnir, D. (1996). Apparent fractality emerging from models of random distributions. *Physical Review E*, 53:3342–3358.

Liebovitch, L. S. and Toth, T. (1989). A fast algorithm to determine fractal dimensions by box counting. *Physics Letters A*, 141(8-9):386–390.

Mandelbrot, B. B. (1982). *The Fractal Geometry of Nature*. Freeman, New York.

Mandelbrot, B. B. (1998). Is nature fractal? *Science*, 279:783–784.

McCorduck, P. (1991). *AARON's Code: Meta-Art, Artificial Intelligence and the Work of Harold Cohen*. Freeman, New York.

Mondada, F., Franzi, E., and Ienne, P. (1993). Mobile robot miniaturisation: A tool for investigation in control algorithms. In *Proceedings of the Third International Symposium on Experimental Robotics*, pages 501 – 503, Berlin. Springer Verlag.

Nolfi, S. (2000). *Evorobot 1.1*. Institute of Cognitive Sciences and Technologies. http://gral.ip.rm.cnr.it/evorobot/simulator.html.

Plotnick, R. E., Gardner, R. H., and O'Neill, V. O. (1993). Lacunarity indices as measures of landscape texture. *Landscape Ecology*, 8(3):201–211.

Sarraille, J. J. and Myers, L. S. (1994). FD3: A program for measuring fractal dimension. *Educational and Psychological Measurement*, 54(1):94–97.

Sibley, F. (2001). Particularity, art and evaluation. In Benson, J., Redfern, B., and Cox, R., editors, *Approach to Aesthetics: Collected Papers on Philosophical Aesthetics, Frank Sibley*.

Spehar, B., Clifford, C. W. G., Newell, B. R., and Taylor, R. P. (2003). Universal aesthetic of fractals. *Computers and Graphics*, 27:813–820.

Creative Ecosystems

Jon McCormack
Centre for Electronic Media Art
Clayton School of IT, Monash Univeristy
Clayton 3800, Australia
Jon.McCormack@infotech.monash.edu.au

Abstract

This paper addresses problems in computational creative discovery, either autonomous or in synergetic tandem with humans. A computer program generates output as a combination of base primitives whose interpretation must lie outside the program itself. Concepts of combinatoric and creative emergence are analysed in relation to creative outputs being novel and appropriate combinations of base primitives, with the conclusion that the choice of the generative process that builds and combines the primitives is of high importance. The generalised concept of an artificial ecosystem, which adapts concepts and processes from a biological ecosystem at a metaphoric level, is an appropriate generative system for creative discovery. The fundamental properties of artificial ecosystems are discussed and examples given in two different creative problem domains. Systems are implemented as pure simulation, and where the ecosystem concept is expanded to include real environments and people as ecosystem components, offer an alternative to the 'software tool' approach of conventional creative software.

Keywords: Artificial ecosystems, Combinationalism, Emergence.

"Theories are important and indispensable because without them we could not orientate ourselves in the world — we could not live. Even our observations are interpreted with their help."
— Karl Popper, *The Myth of the Framework*

1 Introduction

We are interested in problems of *computational creative discovery* where computer processes assist in enhancing human creativity or may autonomously exhibit creative behaviour independently. The intention is to develop ways of working with technology that achieve creative possibilities unattainable from any existing software tools or methods. These goals will be addressed here in the context of artistic creation, however the results may be applicable to many forms of creative discovery.

Darwinian evolution has been described as the only theory with the "explanatory power for the design and function of living systems…accounting for the amazing diversity and astonishing complexity of life" (Nowak, 2006). Evolutionary synthesis is a process capable of generating unprecedented novelty, i.e. *it is creative*. It has been able to create things like prokaryotes, eukaryotes, higher multicellularity and language through a non-teleological process of replication and selection. We would like to adapt, on a metaphoric level, the mechanisms of biological evolution in order to develop new approaches to computational creativity. In Biology, the physical processes of replication and selection take place in an environment, populated by species that interact with and modify this environment, i.e. an ecosystem. Processes from biological ecosystems serve as inspiration for computational artificial ecosystems. The aim is to structure these artificial ecosystems in such a way that they exhibit novel discovery in a creative context rather than a biological one.

We consider creativity in terms that it involves the generation of something *novel and appropriate* (i.e. unexpected, valuable) to the particular aesthetic domain. Van Langen et. al. conclude the necessary conditions for any artificial creative system must be the ability to interact with its environment, learn, and self-organise (van Langen et al., 2004). In this paper, the aim is for creative discovery by machines, or humans and machines working synergistically, rather than a computational model of human creativity or knowledge-based models for a particular domain.

Before looking at how artificial ecosystem concepts can be used as processes for creative discovery, the next section examines how such processes fit into computational creative discovery in general.

2 Combinationalism

A major controversy regarding computational creativity relates to the concept of 'combinationalism': the understanding that "creativity is the creative combination or recombination of previously existing elements" (Dartnall, 2002). This understanding is based on the intuition that one cannot create something new from nothing, hence we require a "combination or recombination of what we already had" — the opposing view being that creativity begins with knowledge, skill and abilities, and emerges from these faculties through interaction with the environment. The challenge is to account for how these cognitive properties give rise to creative output (McCormack, 2005b).

Clearly, many creative outputs are indeed a combination of basic primitives organised in a new way. Let us consider an arbitrary system that generates some creative output from a þxed set of n distinct primitives (basic building blocks, fundamental units). We will call this set of n distinct primitives V, i.e.: $V = \{p_1, p_2, \ldots, p_n\}$. A generative process, G selects elements from V to make $S \in V^r$, an output composed by some permutation of primitives from V. We will assume that:

- the ordering of primitives in S is important;
- repetitions of primitives are permitted;
- The size of S is þxed[1] and $|S| = r$, where $r > 0$.

The process of generating S from V by G is denoted:

$$V \xrightarrow{G} S$$

Denote each speciþc possibil $S_i, i = 1, 2, \ldots, n^r$ (since there are n^r possibilities for S) and $Q \in S^* = \{S_1, S_2, \cdots, S_{n^r}\}$ the set of all possible outputs. Further, let us deþne $Q_G \subseteq Q$ the set of all outputs generated by G. The *conceptual space*, C, is deþned as the b [illegible] nitives, V and the rules for combining them, i.e.: $C = \langle V, G \rangle$.

As a simple example, let us suppose V is a set of musical notes, i.e. $V = \{A, B, C, D, E, F, G\}$ and $r = 12$, so each S is a 12 note melody composed from the notes in V. In this case $n^r = 13,841,287,201$. Clearly, for non-trivial problems the number of possibilities for S is very large, in many cases beyond astronomical proportions such as the estimated number of particles in the universe.

This vast space of potential combinatorial possibilities for S illustrates why such systems are said to display *combinatoric emergence,* that is, conþgurations generated by G appear to express new properties or structures not found in the individual primitive components p. Note that such new properties or structures are generally *observed*, not deþned quantitatively (Baas, 1994; Dorin and McCormack, 2002).

While the potential output generated by G may be vast, any individual output S can *only* be composed of elements from V. In the case of our musical example, we could generate a large number of melodies from V, but none of those melodies could contain the note C♯, for example, because it is not a member of V.

2.1 Creative Emergence

In the case of what is termed *creative emergence*, it is proposed that fundamentally new primitives enter the system, opening up a new set of possibilities that were not previously possible (Cariani, 1991 terms, this process modiþes the c

$$C \rightsquigarrow C^{\Lambda}$$

Where C^{Λ} is the new conceptual space. (2004), in an analogy with letter ative emergence "involves expanding the alphabet of letters by transforming the underlying generative system as well as combining the letters into new words" (Fig. 1). In the terminology used in this paper, creative emergence can introduce new members into V, i.e.: $V \Rightarrow V^{\Lambda}$. The introduction of new primitives in V would by necessity involve some transformation of G, since by deþnit G only knows how to generate things from the original V.

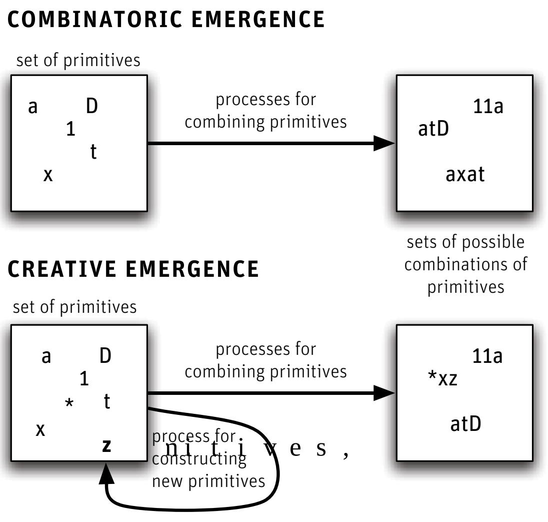

COMBINATORIC EMERGENCE
set of primitives
a D 1 t x
processes for combining primitives
11a atD axat
sets of possible combinations of primitives
CREATIVE EMERGENCE
set of primitives
a D 1 * t x z
processes for combining primitives
11a *xz atD
process for constructing new primitives

Figure 1: Combinatoric and creative emergence (redrawn from (Bird, 2004))

A computational system that combines primitives must provide a semantic interpretation for the members of V. For example, the symbol 'A' must be interpreted as a musical note before it can represent music. It is easy to generate additional primitive *symbols* that can be added to V, but seemingly impossible to computationally discover new interpretations for those symbols, because the interpretation of those symbols is done outside of the software itself (by a listener in the case of a musical example).

There are two conclusions to be drawn from this discussion. The þrst, rather obvio binatorial system, you will only get combinations of the base primitives for which you provide an *a priori* interpretation. The knowledge of how to interpret symbols is provided by the programmer, not the program (the computer can only differentiate one symbol from another).

The second point is that a combinatorial approach is still a useful one if we get our base primitives right. As we have seen, the scope of possibilities is very large in any practicable system. Composers, for example, seem in the main content with composing f primitives. Architects can design great architecture from a þxed set of building materials. made by combining pixels in the right order.

[1]Arbitrary size outputs are possible by incorporating an empty primitive into V, i.e. $V \cup \{\varnothing\}$.

3 Generative processes for Creative Discovery

At this point, we have said nothing about the quality or utility of G and the output it generates. Having a vast range of possibilities in a combinatorial system represents only a *potential* for actually finding good combinations. It is trivial to construct a generative process, G that can generate all the possible members of Q (i.e. $Q_G = Q$). Each combination S_i generated by G will be new, so finding novelty is not the problem, it is finding *appropriate* novelty. The exponential expansion of possibilities, dependent on n and r, means that for any non-trivial system, brute-force methods such as an iterative or random search will not be practical.

Ideally, we would like a generative process that finds the creatively interesting combinations and avoids the uninteresting ones. For many domains the proportion of what we might be inclined to call "interesting" is likely to be extreemly small. Randomly sampling individual members from Q is not, in general, a useful strategy for finding the appropriate members of that set.

If our approach is to relegate creative discovery to being a search or optimisation problem, then a number of general algorithms already exist for this task, e.g. (Michalewicz and Fogel, 1999). One popular choice has been the use of Evolutionary Computing (EC) methods, such as genetic algorithms, evolution strategies or genetic programming.

Standard EC methods require an explicit evaluation of *fitness*, that is a comparative ranking between possible solutions in order to determine the composition of the population for the next generation. For creative discovery, this is a difficult problem for two reasons:

- evaluation of the quality of creative output is highly subjective and context dependent, relying on much domain specific knowledge that is difficult to quantify;
- the type of knowledge and evaluation necessary depends specifically on the creative task or activity begin simulated, i.e. it is difficult to generalise or abstract.

It is for these reasons (and many others) that machine representable fitness functions for "creativity", or "aesthetics" have largely unsuccessful (though not for want of trying, e.g. (Birkhoff, 1933)).

Evaluation of subjective criteria is relatively easy for humans, so a natural approach incorporate human evaluation of fitness into the algorithm. *Interactive Evolution* (also know as *aesthetic selection* or *aesthetic evolution*) have found wide application and popularity for a variety of problems in creative discovery (Takagi, 2001). In this approach, the problem of finding machine-representable fitness functions for aesthetic or subjective properties is circumvented in favour of human fitness evaluation and ranking. While this is a popular method, it is not without significant problems (Dorin, 2001; McCormack, 2005b). These problems include: difficulty in fine-grained evaluation; limited population sizes; slow evaluation times; poor balancing between exploration and exploitation (one of the GA's main benefits as a search method (Eiben and Smith, 2003, p. 29)).

The central question addressed by this paper, then, is this: in a combinatorial system, how can we search and optimise using EC techniques without an explicit fitness evaluation, either by human or machine? That is, what kinds of processes, G are best suited to creative discovery from a combinatorial system? The answer proposed here is through the use of an *artificial ecosystem* approach. This approach is detailed in the following sections.

4 Artificial Ecosystems

The design of environments from which creative behaviour is expected to emerge is at least as important as the design of the individuals who are expected to evolve this behaviour. The *Artificial Ecosystem* as a generalised evolutionary approach for creative discovery. Natural ecosystems exhibit a vast array of complex phenomena, including homeostasis, food-webs, wide causal dependencies and feedback loops, even (controversially) evolution at the ecosystem level (Swenson et al., 2000). Species within the ecosystem compete for resources in order to survive and reproduce. Typical co-operative and competitive evolutionary strategies are observed, such as mutualism, symbiosis, predation and parasitism. To be glib, it could be said that the ecosystem has a lot of interesting features going for it. We would like to harness some of these features for the purposes of *creative discovery* — the discovery of novelty in a system without explicit teleology.

The concept of an artificial ecosystem used here is formative and based on abstractions of selected processes found in biology. We are interested in developing general algorithms for creative discovery. These algorithms are based on dynamic evolutionary processes observed in biological ecosystems. Just as genetic algorithms are not a simulation of natural selection, the artificial ecosystem algorithms presented here are not intended to simulate real biological ecosystems. The ecosystem is viewed as a dynamic, complex system, essential for selection and a driving force behind biological novelty when established with the appropriate conditions. We would like to harness the novel potential of ecosystem processes at a metaphoric level and apply them to creative processes of interest to humans.

4.1 Simulated Ecosystem Studies

Simulated artificial ecosystems have been well studied in the sciences. A number of artificial life models employ the concept of an abstract or simplified ecosystem. This concept of the artificial ecosystem was introduced in (Conrad and Pattee, 1970). A population of independent software agents interact within a programmer-specified artificial physics and chemistry. Agent interaction is simplistically analogous to that which occurs in a real ecosystem. Agents must gain sufficient resources from their environment in order to survive and reproduce. Typically, a number of successful survival strategies will emerge (niches) often with inter-dependencies between individual species (e.g. symbiosis and parasitism). Similar artificial ecosystem methods have been useful in modelling problems in economics (Arthur et al., 1997), ecology (Mitchell and Taylor, 1999) and social science (Epstein and Axtell, 1996).

The majority of such systems focus on single-niche, homogeneous environments, and operate at evolutionary time-scales, simulating the evolution a single species over time. This focus, and the use of minimal, broad assumptions is primarily for the purposes of verification and validation (Adami, 2002). Artificial life agents adapt their behaviour through an evolutionary process to best fit their (typically homogeneous) environment.

Ecological models, on the other hand, tend to operate on far smaller time scales, simulating periods typically ranging from hours to several decades, with a focus on *fitness seeking* through organisational changes or behavioural adaptation of an individual species. This level of simulation reflects the practical questions asked by ecologists in relation to real ecosystems, whereas artificial life research tends to focus on abstract evolutionary dynamics. Important to both styles of investigation is the emergence of macro phenomena or properties from micro interactions. The micro interactions (typically interacting agents) being formally specified in the model; the macro properties an emergent outcome of the simulation.

4.2 Processes for Artificial Ecosystems

In many artificial ecosystem models, the designers of the model are driven by specific applications or outcomes, so the mechanisms, abstractions and terminology differ between systems. This section attempts to define both properties and concepts for general artificial ecosystems. They are positioned at a "middle level" of abstraction: for example an individual is an indivisible unit, it is not represented as a combination of self-organising sub-units, even though this might be possible. In any agent or individual-based model there is always a conflicting tension between model complexity, model validation and simulation outcomes. In contrast to ecological models, the focus of creative discovery is on the suitability and sophistication of creative outcomes, not the verification of models with empirical data or their validation in terms of answering questions not explicit in the original model (Grimm and Railsback, 2005). This allows us some creative licence in our interpretation, but we would still hope for some (at least) semi-formal validation of any general ecosystem models for creative discovery.

While not an essential characteristic of ecosystem models, the use of evolution and the operation on evolutionary time scales is an assumption of the ecosystem models proposed here. This does not preclude the possibility of the model operating at other time scales.

The basic concepts and processes for artificial ecosystems are:

- the concepts of *genotype* and *phenotype* as used in standard EC algorithms. A genotype undergoes a process of *translation* to the phenotype. The genotype and phenotype form the basis of an *individual* in the model;
- a collection of individuals represent a *species* and the system may potentially accommodate multiple, interacting species;
- spatial distribution and (optionally) movement of individuals;
- the ability of individuals to modify and change their environment (either directly or indirectly as a result of their development within, and interaction with, the environment);
- the concept of individual *health* as an abstract scalar measure of an individual's success in surviving within its environment over its lifetime;
- the concept of an individual *life-cycle*, in that an individual undergoes stages of development that may affect its properties, physical interaction and behaviour;
- the concept of an *environment* as a physical model with consistent physical rules on interaction and causality between the elements of the environment;
- an *energy-metabolism resource model*, which describes the process for converting energy into resources that may be utilised by species in the environment to perform *actions* (including the production of resources).

For populations to evolve, there must be some kind of *selection pressure* that implicitly gives some species a higher reproduction rate than others, creating an implicit measure of fitness (Nowak, 2006, Chapter 2). Let us assume any given environment has finite resources and a total population carrying capacity, κ. Species compete for finite resources. These resources are used by individuals to better their reproductive success, until the total population reaches κ. Hence, those able to discover successful strategies for efficiently exploiting those resources are able to reproduce at a higher rate, dominating the population. In contrast to EAs with explicit fitness functions, selection is implicit: successful strategies (individuals) emerge in response to the challenges set by the environment. Moreover, in locating and processing resources, species may alter the environment itself. In this case, adaptation is a dynamic process involving feedback loops and possibly delicate balances.

Individuals maintain a scalar measure of "health" which indicates the success of the individual during its lifetime. This is roughly akin to a fitness measure in traditional EC algorithms. If the health level of an individual falls to zero, the individual dies and is removed from the population (normally returning its resources to the environment). Health is normally affected by the individual's ability to acquire resources from the environment (which may include other individuals). Other internal factors, such as age, may also change an individual's health measure.

In the context of problem solving, individual species may represent competing or co-operating parts of a global solution. This is highly suitable when many different combinations of components may form equally good solutions (e.g. notes or phrases forming a musical composition). When using standard EC methods for search or optimisation, the challenge faced is in choosing appropriate

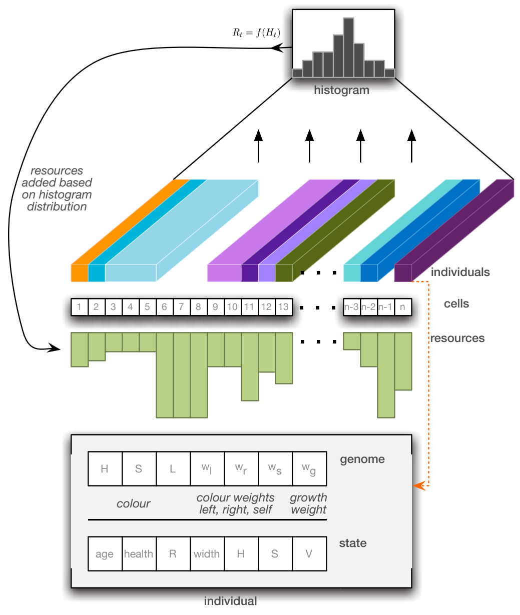


Figure 2: Schematic overview of *Colourfield*

genotype representations, s[illegible]ess functions. The challenge fo[illegible] in the design of environments and the interaction of species within them.

An example of a simple artificial ecosystem model follows.

4.3 Colourfield

Colourfield is a simple one-species ecosystem of colour patterns. It consists of a one-dimensional discrete world of fixed width, populated by i[illegible] (Figure 2). Each space in the world is called a *cell* and may be occupied by at most one individual. Individuals occupy one or more cells and are represented visually as lines of colour. A population of individuals pr[illegible] colours.

An individual's genome is a [illegible] numbers representing: the natural colour (hue, saturation, lightness: HSL); propensity to change to the natural colour, and to the colour of the individual to the left and right of this individual (a normalised weight); propensity to grow into empty neighbour i[illegible] the population maintains a separate *state*, which consists of: the age of the agent, health, current resources held, number of cells currently occupied, and current colour.

All individuals begin with no colour (black) and attempt to acquire resources to reach their target colour (a weighted sum, as determined by the genome, of the natural colour and the current colours of neighbours). Resources are required to change and maintain a particular colour, proportionate to the rate of change. If a neighbouring cell is empty, the individual may "grow" into that cell, the propensity to grow determined by the genome. The more cells occupied, the more resources are required to change colour, but the greater the contribution to the overall *colour histogram* of the world (detailed shortly).

Let the current colour of individual i in RGB colour space be the vector $C_i = (r_i, g_i, b_i)$ and the width w_i. The resources required by the individual are:

$$r_i = w_i^2 \left(k_0 + k_1 \log \left(\frac{d||C_i||}{dt} \right) \right) + k_2 \frac{dw_i}{dt},$$

where k_0, $k_1$and k_2 are constants.

Individuals receive resources from the environment via a feedback process based on the composition of the world. At each timestep, a histogram of chroma and intensity values for the world is built. This histogram, H_t is used as a basis for delivering resources to the world. A total resource R_t for the timestep t, is calculated via a function $f : \mathbb{R}^n \to \mathbb{R}$:

$$R_t = f(H_t)$$

and then distributed equally to all the cells in the world, e.g.:

$$r_{k,t+1} = r_{k,t} + \frac{R_t}{n}, \qquad k = 1, 2, \ldots, n$$

where n is the size of the world. Individuals that occupy more cells therefore receive a greater amount of resources, as they make a greater contribution to the histogram.

A number of different versions of the function f have been tested. These include: favouring chroma values with peaks at equal division, maximising chroma or intensity variation; matching a normal distribution; matching histograms based on paintings recognised for their skilful use of colour.

Given sufficient resources, an[illegible] "growth" an individual may reach its desired colour and width (which may be dependent on the individual's neighbour states). At this time, it may choose to reproduce, either by crossover with an immediate neighbour, or — if there are no neighbours — by mutation. In the case of two immediate neighbours, the mating partner is selected with probability weighted to the norm[illegible] between the colour of the individual and its neighbours, so individuals are more likely to mate with others who produce colours similar to themselves. Offspring are placed in the nearest empty cell, or if none exists, they replace parent. If at any time there are insuf[illegible] is unable to maintain its target colour, causing it to fade and eventually die.

Over time, the system evolves to maximise the production of resources according to the composition of the histogram, which is determined by the size and colour [illegible] of all the individuals in the world. The system exhibits novel colour patterns with patterns of stasis followed by large-scale change as new optima [illegible] covered. Due to the configuratio[illegible] *Colourfield* exhibits classic ecosystem phenomena such as parasitism (a rogue colour contributing little to resource production but "feeding off" other resource producing colours) and mutualism (co-operative combinations of colours mutually contributing to high resource production).

Colourfield is a simple experiment in adapting ecosystem concepts to a simple creative system. It demonstrates creative discovery in a limited domain (creative relationships between fields of colour).

4.4 Tools and Ecosystems

The concept of an ecosystem as a mechanism for creative discovery is not limited to the simulation of ecosystems within the computer. In a creative context it is useful to consider human-machine interaction as forming an ecosystem, replacing the concept of machine as creative tool. This discussion is similar to that used by Di Scipio (2003) and the approach used in the design of the *Eden* system, discussed in Section 4.5.

Humans have always worked with tools. Physical tools are useful because: *(i)* they enable a manipulation of the environment (a chisel sculpts wood); *(ii)* their constraints focus the user to their proper function (a pencil is used for drawing on surfaces); and *(iii)* their organisation encompasses knowledge (we cannot imagine in our mind the correct positioning of a slide rule to evaluate the multiplication of two numbers, yet by physically using a real slide rule it is easy).

Today, computer use is widespread in many areas of creative production, but this use is almost exclusively in the role of “computer as a tool”. Moreover, many of the metaphors used by software tools borrow from physical counterparts or historical lineage (e.g. Adobe Photoshop is a “digital darkroom”, Paint programs use a “virtual paint brush”, etc.). Often these metaphors are poorly translated or simply lack the physicality of their real counterparts (playing a “virtual piano” is just not as good as the real thing).

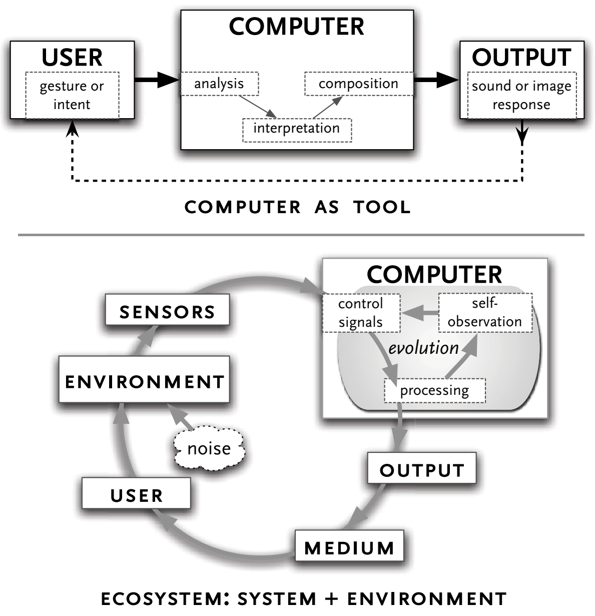


Figure 3: Computer use as a tool (top) and as part of an ecosystem (bottom)

The ecosystem approach does not conceptualise the machine as an object. Rather, the *processes*, both internal and external, are conceived as interdependent, connected *components*, which self-organise into a *system*. Components innately seek to replicate themselves within a noisy environment. With a limited carrying capacity, those components best able to þt the enviro
ulation.

When the system interacts with the environment, it forms an ecosystem. The environment is the medium in which the system develops, and in a creative context may be the creative medium itself (e.g. sound, light, 3D form, and so on). In this mode of working, interdependent processes form an evolutionary, dynamical system, with adaptive behaviour to environmental conditions including the ability to interfere with, and modify, the environment. The machine becomes a synergistic partner in a collaborative creative process, as opposed to a passive tool manipulated by a user. As shown in Fig. 3, the computer, the physical environment and the user all form part of a coupled feedback system.

The powerful properties of tools outlined above are still preserved in the ecosystems scenario, along with additional features not normally associated with the human creative use of tools:

1. *Manipulation of the environment:* components are able to manipulate their environment, moreover due to the recursive coupling (Ash
tem and environment we gain additional properties such as homeostasis (the ability for self-maintenance of particular dynamic conþgur
ternal conditions) and system ‘memory’ through environmental modiþcation.
2. *Constraints are created by the environment:* evolutionary adaptations are þt n
novel solutions imposed by the constraints, not determined by explicit þtness fu
with conventional EC methods.
3. *Organisation encompasses knowledge:* the dynamic conþguration of system compon
knowledge of the system. As th
dynamic and adaptive, the system is able to ‘learn’.

We are interested in new properties and interactions being indirectly implemented: arising as emergent by-products of carefully designed interdependencies between system components.

There are three important considerations in this interactive ecosystem approach to creativity: *(i)* the design of the individual system components and their interdependencies; *(ii)* the metaphors used in interpreting the function of components and their dependencies; and *(iii)* the composition of the environment in which the system interacts. A careful analysis of these considerations remains on-going research.

4.5 Eden: an evolutionary sonic c

Eden is a artwork installation that makes extensive use of the concepts discussed in this paper. The details presented here focus on the ecosystem aspects of the work. For detailed technical descriptions, see (McCormack, 2001, 2005a).

The work consists of a complex a
running in real-time on a two-dimensional lattice of cells. This world is projected into a three-dimensional environment, approximately 6m x 6m (see F

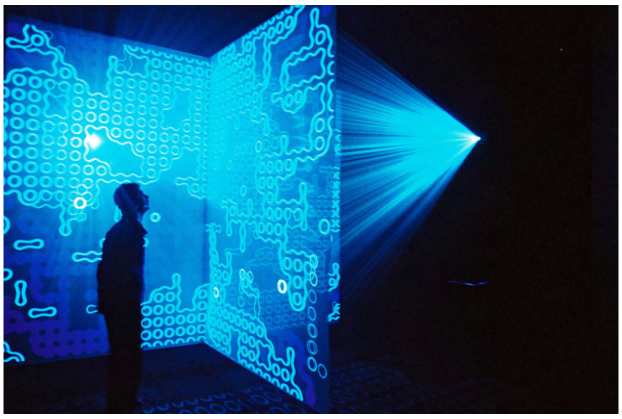


Figure 4: Installation view of *Eden*

consists of three basic types of matter: rocks, biomass, and evolving agents. If a rock occupies a cell, agents or biomass may not. Agents attempting to move into a cell occupied by a rock will "feel" pain and suffer energy loss.

Biomass provides a food source for the agents. Biomass is modelled on an extended *Daisworld* model (Lenton and Lovelock, 2001), with growth rate, β_i for individual biomass element i, a Gaussian function of local temperature at the location (x, y) of the element, T_x, y:

$$\beta_i = e^{-0.01(22.5 - T_x, y)^2}.$$

The *Eden* world exists on an imaginary, Earth-like planet, orbiting a sun with a period of 600 days. The orbit eccentricity and polar orientation result in seasonal variations of temperature, thus affecting biomass growth. As with Lenton and Lovelock's model, the system exhibits self-regulation and stability under a range of conditions. However, overpopulation by agents may reduce biomass to negligible levels, resulting in a temperature increase. The increased temperature lowers the growth rate of the biomass, leading to agent extinction and a dead planet. The system detects such conditions, at which time the planet is "rebooted" to initial conditions and a fresh batch of agents and biomass seeded into the world.

Agents are oriented, omnivorous, autonomous, mobile entities with a collection of sensors and actuators controlled by a learning system, based on classifier systems (a version of Wilson's XCS (Wilson, 1999)). Agents are able to metabolise biomass into energy, which is required to perform *actions* via the agent's actuators. Possible actions include: eating, resting, moving, turning left or right, singing, attacking whatever occupies the cell in front of the agent, mating. The energy cost of these actions varies according to the action (attacking costs more energy than resting, for example), and to physical factors, such as the mass of the agent (mass also increases the power of attacking — a big, heavy agent is more likely to injure or kill a smaller agent). If an agent's energy (health) level falls to 0, the agent dies. Dead agents may be eaten by other agents for a certain time period following death.

Agent sensors are both internal (enabling introspection) and external (enabling sensation of the environment). They include: sensation of cell contents within the Moore neighbourhood of the agent; sound intensity and frequency arriving at the agent's location according to a simple physical model; introspection of pain; introspection of low energy (health). The LCS evolves sets of rules based on past experience and performance of successful rules. At regular periods the agent's health and resource acquisition differentials are examined and a credit or penalty is provided to those rules used since the previous evaluation. A positive differential pays credit proportional to its magnitude, likewise a negative differential penalises. Successful rules gain credit and so are more likely to be selected in the future. Rules that consistently receive penalty are eventually removed.

Rules evolve during an agent's lifetime, with a penalty imposed on energy for large rule sets to encourage efficiency. Two agents may mate — the resultant offspring inherit the most successful rules of their parents, hence the system uses *Lamarkian evolution*.

The *Eden* environment is visualised and sonified in the installation space. The two-dimensional world is projected onto two translucent screens, configured in an 'X' shape. This enables people experiencing the work to move freely around the screens at close range, examining details of the world as it updates in realtime. The sounds made by the agents are spatially mapped to four speakers located at the two corners of each screen. This rough spatialisation permits the listener to approximately locate the sound source within the *Eden* world. The bandwidth devoted to sound is much higher than any other sensory information used by the agent. Agents are able to differentiate and make sound over a range of frequency bands, giving rich opportunities for the use of sound in an ecosystem context.

In addition to the internal ecosystem model, the *Eden* world is also connected to the physical world of the installation space via an infrared video camera which tracks the presence and motion of people looking and listening to the artwork[2]. The presence of people in the installation space influences the growth of biomass in the virtual space. The longer people spend with the work, the more food is likely to grow in the virtual environment. The rationale for this is driven by the idea that the more interesting people find the work, the longer they will stay. If they find the work uninteresting, they will not spend much time with it. A good way to maintain people's interest is to produce sounds, moreover, *interesting, changing* sounds.

Over time, the agents evolve to make complex sounds in order to maintain their food supply. The agents have no specific knowledge of people in the environment, however, by making interesting combinations of sounds they attract and maintain the interest of the human audience in the environment[3]. This interest translates to a more stable supply of food, hence improving chances of survival in the environment. Therefore, *Eden* is a symbiotic ecosystem, which includes the human audience experiencing the work.

[2]The original version of the work used infrared distance sensors.

[3]When shown in a gallery environment, it is important to remember to compensate for opening hours, otherwise the population dies out each night when the gallery is closed!

5 Conclusions

In contrast with previous attempts to model creativity, which have applied psychological, cognitive, or knowledge-based models of human creativity, the ecosystem approach sees creativity as an emergent phenomenon of dynamic interaction between interconnected, self-organising components and their environment. These components and their environment may be internal to computer simulation (as in the *Colourfield* system) or part of a system that incorporates humans and the physical environment (as with the *Eden* system).

Combinatorial systems do not practically impose the limitations that might be suggested by the opposing concepts of combinatoric and creative emergence. Necessarily, all base primitives must contain an interpretation that lies outside the software itself. What is important is the process used to derive a creative result from a set of base primitives. The goal is to enable the synergistic exploration of new conceptual spaces in creative partnership with the machine. In the artificial ecosystem approach, this can be achieved by developing a formal understanding of the appropriate design of components, their interconnections, and the environment in which they operate.

References

Adami, C. (2002). Ab initio modeling of ecosystems with artificial life. *Natural Resource Modeling*, 15:133–146.

Arthur, W. B., Durlauf, S., and Lane, D. A., editors (1997). *The economy as an evolving complex system II*. Addison-Wesley, Reading, MA.

Ashby, W. R. (1952). *Design for a Brain*. Chapman & Hall, London.

Baas, N. A. (1994). Emergence, Hierarchies and Hyperstructures. In Langton, C. G., editor: *Artificial Life III*, 515–537. Addison-Wesley, Reading, MA.

Bird, J. (2004). Containing Reality: Epistemological Issues in Generative Art and Science. In *Impossible Nature: the art of Jon McCormack,* 40–53. Australian Centre for the Moving Image, Melbourne.

Birkhoff, G. D. (1933). *Aesthetic Measure*. Harvard University Press, Cambridge, MA.

Cariani, P. (1991). Emergence and Artificial Life. In Langton, C. G. et. al., editors: *Artificial Life II, SFI Studies in the Sciences of Complexity,* 775–797. Addison-Wesley, Redwood City, CA.

Cariani, P. (1997). Emergence of new signal-primitives in neural systems. *Intellectica*, 2:95–143.

Conrad, M. and Pattee, H. H. (1970). Evolution experiments with an artificial ecosystem. *Journal of Theoretical Biology*, 28:393.

Dartnall, T., editor (2002). *Creativity, Cognition, and Knowledge: An Interaction*. Praeger, Westport, Connecticut.

Di Scipio, A. (2003). 'Sound is the interface': from interactive to ecosystemic signal processing. *Organised Sound*, 8(3):269–277.

Dorin, A. (2001). Aesthetic fitness and artificial evolution for the selection of imagery from the mythical infinite library. In Kelemen, J. and Sosík, P., editors, *Advances in Artificial Life*, LNAI 2159, 659–668. Springer-Verlag, Berlin.

Dorin, A. and McCormack, J. (2002). Self-Assembling Dynamical Hierarchies. In Standish, R. K., et. al. editors: *Artificial Life VIII: Proceedings of the Eight International Conference on Artificial Life*, 423–428. MIT Press, Cambridge, MA.

Eiben, A. E. and Smith, J. E. (2003). *Introduction to Evolutionary Computing*. Natural Computing Series. Springer, Berlin.

Epstein, J. M. and Axtell, R. (1996). *Growing Artificial Societies*. MIT Press, Cambridge, MA.

Grimm, V. and Railsback, S. F. (2005). *Individual-based Modeling and Ecology*. Princeton Series in Theoretical and Computational Biology. Princeton University Press.

Lenton, T. M. and Lovelock, J. E. (2001). Daisyworld revisited: quantifying biological effects on planetary self-regulation. *Tellus*, 53B(3):288–305.

McCormack, J. (2001). Eden: An evolutionary sonic ecosystem. In In Kelemen, J. and Sosík, P., editors, *Advances in Artificial Life*, LNAI 2159,133–142. Springer-Verlag, Berlin.

McCormack, J. (2005a). On the Evolution of Sonic Ecosystems. In Adamatzky, A. and Komosinski, M., editors: *Artificial Life Models in Software* 211–230. Springer-Verlag, London.

McCormack, J. (2005b). Open problems in evolutionary music and art. In Rothlauf, F. et. al. editors, *EvoWorkshops*, LNCS 3449, 428–436. Springer, Berlin.

Michalewicz, Z. and Fogel, D. B. (1999). *How to solve it: modern heuristics*. Springer, New York.

Mitchell, M. and Taylor, C. E. (1999). Evolutionary computation: An overview. *Annual Review of Ecology and Systematics*, 30:593–616.

Nowak, M. A. (2006). *Evolutionary Dynamics: exploring the equations of life*. The Bekknap Press of Harvard University Press, Cambridge, MA, and London, England.

Swenson, W., Wilson, D. S., and Elias, R. (2000). Artificial ecosystem selection. *PNAS*, 97(16):9110–9114.

Takagi, H. (2001). Interactive evolutionary computation: Fusion of the capabilities of ec optimization and human evaluation. *Proceedings of the IEEE*, 89:1275–1296.

van Langen, P. H. G., Wijngaards, N. J. E., and Brazier, F. M. T. (2004). Towards designing creative artificial systems. *AIEDAM, Special Issue on Learning and Creativity in Design*, 18(4):217–225. A. H. B. Duffy and F. M. T. Brazier (editors).

Wilson, S. W. (1999). State of XCS classifier system research. Technical report, Concord, MA.

Towards a General Framework for Program Generation in Creative Domains

Marc Hull
Department of Computing
Imperial College
180 Queen's Gate
London
SW7 2RH
mfh@doc.ic.ac.uk

Simon Colton
Department of Computing
Imperial College
180 Queen's Gate
London
SW7 2RH
sgc@doc.ic.ac.uk

Abstract

Choosing an efficient artificial intelligence approach for producing artefacts for a particular creative domain can be a difficult task. Seemingly minor changes to the solution representation and learning parameters can have an unpredictably large impact on the success of the process. A standard approach is to try various different setups in order to investigate their effects and refine the technique over time.

Our aim is to produce a pluggable framework for exploring different representations and learning techniques for creative artefact generation. Here we describe our initial work towards this goal, including how problems are specified to our system in a format that is concise but still able to cover a wide range of domains. We also tackle the general problem of constrained solution generation by bringing information from the constraints into the generation and variation process and we discuss some of the advantages and disadvantages of doing this. Finally, we present initial results of applying our system to the domain of algorithmic art generation, where we have used the framework to code up and test three different representations for producing artwork.

Keywords: Automatic program generation, genetic programming, evolutionary art.

1 Introduction

Finding an efficient approach for producing artefacts in a particular creative domain is often more of an art than a science. Many general artificial intelligence techniques exist that could potentially be used with varying degrees of success, but most are so complex that it can be difficult to tell in advance which will perform better than others. They are also heavily dependent on the problem representation used and a number of other parameters that can greatly affect their performance. Typically this can be alleviated by using knowledge of the problem to predict which search strategies will be most successful. However, in creative domains the problem may not be well-defined enough to be an accurate guide.

Our aim is to build a general, pluggable framework where problems can be expressed using a concise syntax and tested with all of the different artificial intelligence techniques written for the system. In this paper, we address the problem of how the user specifies a search space to our system. To cover a wide range of domains and learning techniques, the representation used must be general enough to cover many different data structures, but not too general as to include invalid solutions in the search space. To achieve this, we opted for a tree-based structure, similar to those used in Genetic Programming, with implicit type rules controlling the shape of the tree and explicit constraints to disallow particular node patterns. In later work, we hope to concentrate on evaluating different search strategies and attempting to define classes of domains for which particular techniques work best.

In section 2 we provide the background to our project by describing existing genetic programming tools and some of their applications to creative artefact generation, against which we compare our approach. In section 3 we provide details of the framework in terms of how we split the specification of problems into three concise parts, which enables quick, flexible prototyping of different representations. In section 4 we describe how solutions are generated via an iterative two-stage process that employs explicit user-given constraints about the nature of the programs to be generated in order to avoid producing invalid solutions.

To demonstrate the potential of the framework for productive generation of programs, we have implemented an evolutionary search mechanism where the user acts as a fitness function (a standard approach), and we have applied it to algorithmic art generation. Using the concise problem specification syntax we were able to quickly prototype three different representations for generating programs that produce artwork when executed, namely colour based, particle based and image based approaches. Finally, in sections 6 and 7, we describe the future work planned for the framework and summarise our conclusions drawn so far.

2 Background

Genetic Programming was first popularised by Koza's 1992 book (Koza, 1992) as a way to find programs that optimise a measure of fitness. By manipulating variable-length parse trees that represent Turing-complete programs, it is sufficiently general to be able to represent solutions from many other machine learning techniques (Banzhaf et al., 1998). Many tools and libraries exist for using this technique to evolve programs or program fragments to perform particular tasks, e.g. DGPF (Weise and Geihs, 2006), the Genetic Programming Engine[1] and GALib[2]. Typically, these allow the user to specify a representation containing the ingredients (terminals and non-terminals[3]) to be used in the solutions. The user also supplies a fitness function that evaluates each solution, returning a numeric result indicating its suitability for performing the task. To start with, a set of random solutions are generated using the given representation and these are then evaluated by the fitness function to determine which perform better than others. Roughly speaking, the best performing solutions are then combined with one another in an attempt to form new solutions that produce a higher value when evaluated by the fitness function. This process is then repeated until the required minimum fitness value is exceeded, at which point the solution with the highest fitness value is taken as the final solution.

In the pure GP approach, solutions can be constructed using any combination of terminals and non-terminals provided in the representation. However, often this results in solutions being produced that are not valid in the context of the problem. A simple example is a problem that requires a permutation of values, where any solution that contains the same value twice would be invalid.

A standard GP approach would still allow such invalid solutions to exist, but one remedy to this is to apply additional constraints to the representation to explicitly remove certain terminal and non-terminal combinations from the search space. This then leads to the problem of constrained solution generation and variation, where the technique used for producing and altering solutions must take the constraints into account in order to avoid invalid solutions.

Several approaches for handling constraints in evolutionary techniques have been tested, with the most notable being penalty functions, repair functions and decoder functions. Penalty functions (Baeck et al., 1995) evaluate solutions against each constraint individually and subtract a value from their fitness for every constraint violated. Repair functions (as used in Michalewicz and Nazhiyath (1995)) allow solutions that violate constraints to be generated, but then modify these solutions in an attempt to find the most similar variant that passes all of the constraints. Decoder functions (as in Gottlieb and Raidl (2000)) are employed during the genotype to phenotype mapping phase and ensure that the solution genotype maps on to a phenotype in which all constraints will always be satisfied.

Constraint handling is particularly appropriate when the solution phenotype is a computer program in a high-level language, since such languages often impose many complex constraints such as scoping rules and type compatibility upon their programs. Strongly Typed Genetic Programming helps to alleviate some of these problems by allowing the representation itself to be typed and then modifying the generation and evolution algorithms to only produce type-safe trees (Montana, 1995). However, this is often achieved by tying the GP system to a particular language, such that the rules of that language are implicit in the representation (as in JGAP[4]).

Genetic Programming has also previously been used to evolve programs that produce artefacts from creative domains such as pictures and music. Machado and Cardoso's NEvAr system (Machado and Cardoso, 2002) is a well-known generator of algorithmic art which evolves an algorithm for setting the red, green and blue colour components of each pixel in an image. Johanson and Poli have applied a similar technique to music generation (Johanson and Poli, 1998). Their system produces programs in a custom language that describes how to play chords and when to pause between notes. Many other systems also use automatic program generation to produce creative artefacts, however a full survey of these is beyond the scope of this paper.

3 Framework Details

Our aim is to provide a framework that is general enough to accept problems from a wide range of different domains, yet concise enough to allow for quick prototyping and easy modification. We have chosen a variable-sized tree representation to encode solutions, similar to those used in Genetic Programming, since this is sufficiently general to also encode representations used in other machine learning approaches. However, an overly general representation would include solutions in the search space that may not be valid solutions to the problem being solved. Hence, the framework must also allow users to easily constrain their representations to remove invalid solutions for specific problems.

To allow users to specialise their representations, we use a combination of node type constraints upon our parse trees and logical constraints for removing unwanted node patterns. The former is based upon the constraints of Montana's Strongly Typed Genetic Programming system (Montana, 1995) and allows the type systems of programming languages to be respected. Meanwhile, the latter allows for constraints over node dependencies to be expressed. This can, for instance, be used to enforce that instances of two node types may only exist together and not independently. We have found this to be particularly useful for expressing relationships between function calls and function declarations when evolving programs.

Finally, we also provide a method for translating the

[1]A genetic programming library for the .NET framework (http://gpe.sourceforge.net/)

[2]A C++ library of genetic algorithm components (http://lancet.mit.edu/ga/)

[3]These are described as terminals and functions in Koza (1992), but we use the term non-terminal here to distinguish from functions in programs.

[4]An open-source, Java-based genetic algorithms package (http://jgap.sourceforge.net)

tree structure used internally to represent solutions into text output, which is analogous to the genotype-phenotype mapping in Genetic Programming. In our experiments, we use this to convert our solutions into programs, scripts or data that can be accepted by other programs. We then use the behaviour of these programs to evaluate the success of the solution.

To keep the roles of specifying the representation, imposing constraints and compiling solutions to text separate, users provide each of these to our system in a separate file. The following subsections explain how these files work in further detail.

3.1 Representation File

Solutions in our current system are represented by trees whose structure is specified by the user in the representation file. At a basic level, this file allows the non-terminal and terminal node types of the tree to be specified, in a similar way to most other Genetic Programming systems. However, these node types are also involved in typing constraints that allow the structure of the trees to be controlled.

These typing constraints allow the terminals and non-terminals of the representation to exist in an inheritance hierarchy, such that groups of node types that are semantically linked (e.g. `True`, `False`, `And`, `Or`) can inherit from a common node supertype (e.g. `Boolean`) that represents this link. Each non-terminal node type then specifies which arcs it has to each of its child nodes, and also inherits any arcs declared in its supertypes. Each arc is also annotated with node types that restrict the nodes that can be children of it. An arc annotated with a node type `X` will only accept child nodes that are instances of the `X` type or any types that inherit from `X`. Additional features such as abstract node types, primitive types and multi-child arcs are also supported but there is insufficient space here to describe them in detail.

The following shows an example of the syntax used to specify a representation for simple numeric expressions.

```
representation NumericExpressions {
  abstract type NumericExpression;
  type Zero : NumericExpression;
  type One : NumericExpression;
  type Two : NumericExpression;
  abstract type BinaryOperator
              : NumericExpression {
    NumericExpression left;
    NumericExpression right;
  };
  type Add : BinaryOperator;
  type Sub : BinaryOperator;
  type Mul : BinaryOperator;
  type Div : BinaryOperator;
};
```

3.2 Constraints File

In addition to the implicit typing constraints provided in the representation file, the user can also specify explicit constraints upon the solution trees in the constraints file. Since the system is not tailored to output in a specific language, this file can be used to add constraints that are specific to the output language for this particular problem. It can also be used to add domain-specific constraints, which in the case of program generation could remove a large proportion of non-compiling and invalid solutions from the search space.

Constraints are currently expressed in a syntax that is based upon first-order logic, but is tailored to expressing conditions about tree structures. The language includes `and`, `or` and `not` operators, which follow their traditional logical semantics, as well as `exists` and `all` operators that have special meanings. In particular, they only match nodes at or within a particular part of the tree, and they can optionally bind these matches to variables that are then used in the evaluation of their sub-expressions.

The following shows how this syntax can be used to express the constraint that `Div` nodes cannot have `Zero` nodes for their `right` children in the representation from Section 3.1.

```
constraint NoDivideByZero {
  all Div in root as divideNode (
    not (
      exists Zero at
      divideNode.right
    )
  )
};
```

3.3 Compiler File

Once a solution tree has been generated, the compiler file is consulted for the transformations required to convert the tree into the specified output language. The user provides these transformations as string templates for each node type which describe how nodes of that type should be represented in the output. The string templates are specified in the Velocity templating language[5], so that references to the compiled output of child nodes are represented by enclosing the child name between `${` and `}` delimiters. This also allows the templates to include control flow constructs like `for` loops over node children.

The following gives the compiler code for translating the numeric expression representation from Section 3.1 into C-style expression syntax.

```
compile Zero [|0|];
compile One  [|1|];
compile Two  [|2|];
compile Add  [|(${left})+(${right})|];
compile Sub  [|(${left})-(${right})|];
compile Mul  [|(${left})*(${right})|];
compile Div  [|(${left})/(${right})|];
```

4 Constraint Handling

In section 2 we highlighted three existing methods for handling constraints in evolution-based systems; penalty functions, repair functions and decoder functions. For our

[5]An open-source, Java-based string template language (http://velocity.apache.org/)

system, we have tried a new approach to constraint handling, in which information concerning the constraints is used to guide the initial process of solution generation, then constraint-aware variation operators are used to produce only valid children.

To guide the generation and variation operators, we use an approach that attempts to determine whether the tree being modified violates the constraints either directly or indirectly. A direct violation is where the nodes in the tree contradict at least one of the constraint conditions, whereas an indirect violation is where a partial tree[6] restricts the possible nodes that can be placed to only those that will contradict at least one of the constraint conditions. To make this problem tractable, the constraint language was restricted to only a small number of operators, which allowed us to hard-code a number of routines that were able to reason about the constraints at the sacrifice of losing Turing-completeness of the language. The following subsections cover the algorithms used for guiding the generation and variation of trees in further detail.

4.1 Solution Generation

Solutions are generated using an iterative two-stage process of checking which possible valid node instantiations can be made and then choosing one based on knowledge of previous good solutions. To start with, the generator component takes a partial tree as input, so to generate a tree from scratch a root node must first be instantiated and passed to the generator. As the first step, the generator sets all unset arcs to point to placeholder nodes and then adds all possible combinations of placeholder nodes and their type-compatible node types to a list of possible choices that can be made. The generation process then proceeds as follows:

- The tree constraints are evaluated with respect to the nodes currently in the tree to produce the constraints that must be satisfied by the remaining nodes to be added.
- Each of the possible choices is checked against the constraints and is removed if they would directly or indirectly violate them.
- If there is a placeholder for which there are no remaining possible choices, the algorithm backtracks.
- Otherwise, one of the choices is picked at random, its node type is instantiated and its corresponding placeholder is replaced with the new node instance. All alternative choices for that placeholder are then removed from the list of possible choices. All children of the new node are set to placeholder nodes and all combinations of the new placeholders and their type-compatible node types are added to the list of possible choices to be made.
- If there are no placeholders in the tree, the algorithm terminates, otherwise it repeats from step one.

[6]A partial solution tree is a tree where some branches end in non-terminal nodes rather than terminal nodes, and so some arcs have yet to be assigned child nodes.

4.2 Solution Improvement

Currently, we use minor variations on traditional GP techniques of crossover and mutation to produce new solutions from previous ones, with the initial population generated using the above algorithm of constrained random generation.

For crossover between two trees, T_1 and T_2, we randomly select a node N_1 from the first tree which determines its crossover point, but then we filter the nodes in the second tree by those that would form a type-compatible tree when interchanged with N_1. A node N_2 is then randomly chosen from this filtered list and the subtree rooted at N_1 in T_1 is swapped with the subtree rooted at N_2 in T_2. The tree constraints are then checked and, if violated, the swap is undone and N_2 is removed from the list of filtered nodes and another node is chosen. If no valid replacement node for N_1 can be found, a new crossover point is chosen in T_1. Mutation of a single tree is handled by randomly selecting a node N_1, removing the subtree rooted at N_1 and then passing the tree to the generator to fill in the gap.

4.3 Violation Detection

In this approach, the ability to reason about the constraints in order to predict which choices would directly or indirectly violate them has a large influence on the performance of the system. Currently, we preprocess both the constraints and the representation in order to build up the following meta-information that can be queried by the system in order to detect whether a violation has occurred:

- The *Must Type Set* of a node type contains itself and all of its supertypes.
- The *Transitive Must Type Set* of a node type is defined as the set of node types that must appear in a subtree rooted at a node of the given type.
- The *May Type Set* of a node type contains itself, all of its supertypes and all of its non-abstract subtypes.
- The *Transitive May Type Set* of a node type is defined as the set of possible node types and supertypes that can appear in a subtree rooted at a node of the given type.
- The *Shortest Terminal Length* of a node type is the minimum number of arcs that must be traversed from nodes of that type before a terminal node is reached.

A fixed set of rules are then used to determine whether a violation has occurred. These rules contain a pattern part that is matched against parts of the constraints and a condition part that, based on the current tree and the results of queries over the meta-data, returns whether or not the constraint can be satisfied. For example, one rule looks for constraints of the form `exists NodeType in Subtree`, where `NodeType` and `Subtree` are variables, and will then check whether `NodeType` is within the May Type Set of any placeholder nodes within `Subtree`. If it is not, then this rule has successfully determined that the constraint can never be satisfied by any complete trees built upon the current partial tree.

4.4 Evaluation

So far, we have tested our constraint handling approach on a number of small examples from very simple constraints, such as asserting that a certain node type must appear in all solutions, to complex ones based on node type co-dependency. Although we do not have enough results to produce a full quantitative analysis of the approach, we have noticed good performance in the face of complex constraints where the space of valid solutions is sparse. Unfortunately, this is often hidden by poorer performance when faced with simple constraints (due to the overhead of the system) or combinations of constraints that are not covered by our reasoning rules. This is partly because, when faced with a problem for which no rules exist, our system degenerates to an exhaustive search of the solution space, which can result in repeatedly taking paths that lead to dead ends.

However, one advantage that our system may have over penalty-based approaches (which we intend to check empirically) is that the destructive effect of mutation and crossover is reduced by guaranteeing that offspring will always be valid solutions. Such destructive effects (when children have lower fitness than their parents) can lead to introns and bloat in members of the population, which can hamper the evolutionary process (Soule and Foster, 1997).

5 Application to Algorithmic Art

To test our system, we prototyped three different problem specifications for generating different types of algorithmic art. The three types that we focused on were:

- Colour-based artwork, where the algorithm used to set the colour of each pixel in the picture is evolved, in a similar vein to NEvAr (Machado and Cardoso, 2002).
- Particle-based artwork, where the algorithm used to set the position and colour of 1000 particles is evolved and the particle trails are plotted over 100 time steps.
- Image-based artwork, where the algorithm used to set the colour of each pixel in an image can also use colour values from a source image.

A different problem specification was written for each of the above types, but each one outputs code in a language based on Processing[7], a scripting language used by graphic artists that is tailored to providing high-level drawing operations. These scripts were then executed to produce the resulting images.

5.1 Colour-Based Artwork

In this representation, the resulting programs loop over all pixels in the output image and set their hue, saturation and value based on some algorithm. The part of the program that loops over all the pixels is constant between solutions, however the algorithms used to set the hue, saturation and value of each pixel can vary. To allow for this, the representation has node types for constructing floating-point expressions which include constants (within the range 0.0 to 1.0 inclusive), simple mathematical functions (add, subtract, multiply, divide, sin, cosine and random) and variables (the x and y position, expressed as screen proportions).

Since the representation is quite restricted, the type system in the representation is enough to ensure that all produced solutions compile. However, there are still a number of compiling solutions that we want to rule out, such as those that cause errors at runtime or produce pictures that we know will be judged badly. To remove these from the search space, we added the following constraints to the constraints file:

- There must be a variable somewhere in the solution tree, where a variable is a reference to the x or y position in the image or a call to random. All solutions that do not contain variables will always produce images in which all pixels have the same colour.
- No constant representing the number one must ever appear as an operand of a multiply expression. Any solution that contains this combination could be simplified and so is redundant.
- All functions must contain at least one variable as one of their operators. This avoids constant subexpressions that may create values outside of the desired range or may be more simply expressed as a single constant.

The compiler file then specifies the mapping between the node types and the corresponding scripting code that draws the image. All of the numerical expression node types map to their expected operators, function calls or variable names, while the root type maps to the code that loops over the image and uses the generated expressions to set the hue, saturation and colour components of each pixel as shown below:

```
compile Main {
  |int width = 500;
  |int height = 500;
  |public void setup() {
  |  size(width, height);
  |  background(
  |    hsv(0.0f, 0.0f, 0.0f)
  |  );
  |  for (float y=0; y<1;
  |       y+=1/(float)height) {
  |    for (float x=0; x<1;
  |         x+=1/(float)width) {
  |      float h = ${hue};
  |      float s = ${saturation};
  |      float v = ${value};
  |      pixel(x, y, hsv(h, s, v));
  |    }
  |  }
  |}
};
```

[7]See http://www.processing.org

Overall, the colour-based artwork problem specification consists of 130 lines of text spread across these three files which describes 29 node types, 5 constraints and 24 compiler rules. With this, we could generate, crossover and mutate solutions using the algorithms described in Section 4.1 and 4.2 and then compile and execute the resulting scripts and inspect the images generated. In figure 1, we present some example images generated using this representation.

5.2 Particle-Based Artwork

The second representation to be tested used a simple particle simulation as a basis for producing artwork. The generated part of the solution is the algorithm used to control the position and colour of 1000 particles over time. The static part of the solution creates the 1000 particles and then plots their trails over 100 time steps. In addition to this, a convolution is applied to the resulting image after each time step, the kernel of which can also vary. This has the result that lines drawn in early time steps will often appear more blurred than those drawn in later time steps, so that an impression of how the simulation has progressed over time can be seen in the resulting image.

The representation used here was very close to the colour-based representation, except with the variables now tracking the position, colour, previous position, time and index of each particle. Where the colour-based representation only evolved three numerical expressions, this representation evolves 12; six to initialise the position and colour of every particle and six more to update the position and colour of every particle in every time step, in addition to three constants that control the background colour of the image.

The constraints upon the representation are also more complex than those for the colour-based artwork, mainly due to the additional variables that are only in scope for particular parts of the program. For example, it makes no sense to reference the time step number or a particle's previous position in its initialisation expressions, so these are explicitly disallowed in the constraints.

Overall, the particle-based artwork problem specification consists of 238 lines of text which describes 44 node types, 6 constraints and 38 compiler rules. In figure 2, we present some example images generated using this representation.

5.3 Image-Based Artwork

The third representation to be tested used an existing image as input and could query this image for its hue, saturation and value components at any point, then use these values in numerical expressions for setting the colour of each pixel in the output image. This allowed it to produce image-filter style images by setting the output pixel colours to some function of the source pixel colours. It could also produce warps of the source image by assigning the output pixels to pixels at different positions in the source image based on some numeric function. Finally, it could also evolve the kernel of a convolution filter to be applied as a post-processing step. The result of this is that a range of images are produced, some of which obviously contain the source image filtered in some way, and others which merely use it as a source of semi-random values.

The constraints for this representation were very similar to those of the colour-based representation, except that additional constraints were added to force all solutions to use the source image colours somewhere in its computation of the output image colours. This ensured that the search space of this representation did not include the search space of the colour-based representation as a subset.

Overall, the image-based artwork problem specification consists of 171 lines of text which describes 34 node types, 6 constraints and 28 compiler rules. In figure 3, we present some example images generated using this representation along with the source image used to produce them.

6 Future Work

In section 5, we used the domain of algorithmic art to test the usage of our framework for creative artefact generation, where we were able to use simple problem specifications to produce artworks of a similar nature to those produced with bespoke systems. However, many of the design decisions made during the development of our system have been motivated by our interest in scaling it up to handle much more complex problems efficiently. We are currently using the system in domains such as interactive art and the generation of simple computer games and have plans for 3D model and landscape generation.

Early results from these domains show that the evaluation of solutions is much more time-consuming than that of the algorithmic art shown here. For interactive domains in particular, the user must often try various input combinations in order to test for a response. We therefore believe that it is important for the system to extract more information from each evaluation. One way to achieve this is to allow the user to drill down into each solution in order to target the specific parts that are performing poorly. The system could then refine these parts separately until they meet the user's satisfaction, when they could be recombined with the rest of the solution.

We also intend to investigate general ways of allowing our system to refine solutions semi-autonomously in order to reduce the number of evaluations performed by the user. This could be done by allowing users to specify their preferences to the system as a fitness function over the phenotype, or a machine learning approach could be used to learn their preferences from the initial evaluations made during each session. By using a logic-based learning method such as Inductive Logic Programming (Muggleton, 1991), this opens up the possibility of the user understanding and altering the learned fitness function.

Finally, we acknowledge the that evaluation of systems and the artefacts they produce is an essential aspect of computational creativity which is missing from the work presented here and we aim to fill this gap. We are already planning a number of studies for assessing both the usability of our system and the appeal of the artefacts that it produces. We are also looking into ways to quantitatively evaluate parts of our system where possible.

7 Conclusions

We have presented the first description of our generic framework for automated program generation, and demonstrated its usage in an evolutionary art setting. We have found that bringing constraint checking into the solution generation process can help weed out systematically poor solutions and produce solutions faster than traditional generate-and-test approaches. As we saw with the application to three separate art generation problems, our framework enables rapid development of program generation systems. This has helped us to quickly prototype, test and refine various representations for different program generation problems.

Our current implementation is lacking in a number of areas that prevent us from applying it to solve more complex problems. Although the use of constraints helps to rule out many bad solutions, we will need to supplement this with more sophisticated constraints and a flexible fitness calculation mechanism. This is because, for more complex representations, we've found our existing constraint language is insufficient for ruling out enough bad solutions to enable convergence on good solutions within a reasonable time. However, this work is ongoing, and we expect to find a number of ways to address these issues in order to allow us to test the system on a wide range of different domains.

Acknowledgements

We would like to thank the anonymous reviewers for their comments which have helped us to improve this paper.

References

Baeck, T., Fogel, D., and Michalewicz, Z. (1995). Penalty functions. *Handbook of Evolutionary Computation*.

Banzhaf, W., Nordin, P., Keller, R. E., and Francone, F. D. (1998). *Genetic Programming: An Introduction*.

Gottlieb, J. and Raidl, G. R. (2000). The effects of locality on the dynamics of decoder-based evolutionary search. In *Proc. of the Genetic and Evolutionary Computation Conference 2000*, pages 283–290.

Johanson, B. and Poli, R. (1998). GP-music: An interactive genetic programming system for music generation with automated fitness raters. In *Genetic Programming 1998: Proc. of the 3rd Annual Conference*, pages 181–186.

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

Machado, P. and Cardoso, A. (2002). All the truth about NEvAr. *Applied Intelligence*, 16:101–118.

Michalewicz, Z. and Nazhiyath, G. (1995). Genocop iii: A co-evolutionary algorithm for numerical optimization problems with nonlinear constraints. In *Proc. of the 2nd IEEE International Conference on Evolutionary Computation*, pages 647–651.

Montana, D. J. (1995). Strongly typed genetic programming. *Journal of Evolutionary Computation*, 3:199–230.

Muggleton, S. (1991). Inductive Logic Programming. *New Generation Computing*, 8(4):295–318.

Soule, T. and Foster, J. A. (1997). Code size and depth flows in genetic programming. In *Proc. of the 2nd Annual Conference on Genetic Programming*, pages 313–320.

Weise, T. and Geihs, K. (2006). DGPF - an adaptable framework for distributed multi-objective search algorithms applied to the genetic programming of sensor networks. In *Proc. of the 2nd International Conference on Bioinspired Optimization Methods and their Application, BIOMA 2006*, pages 157–166.

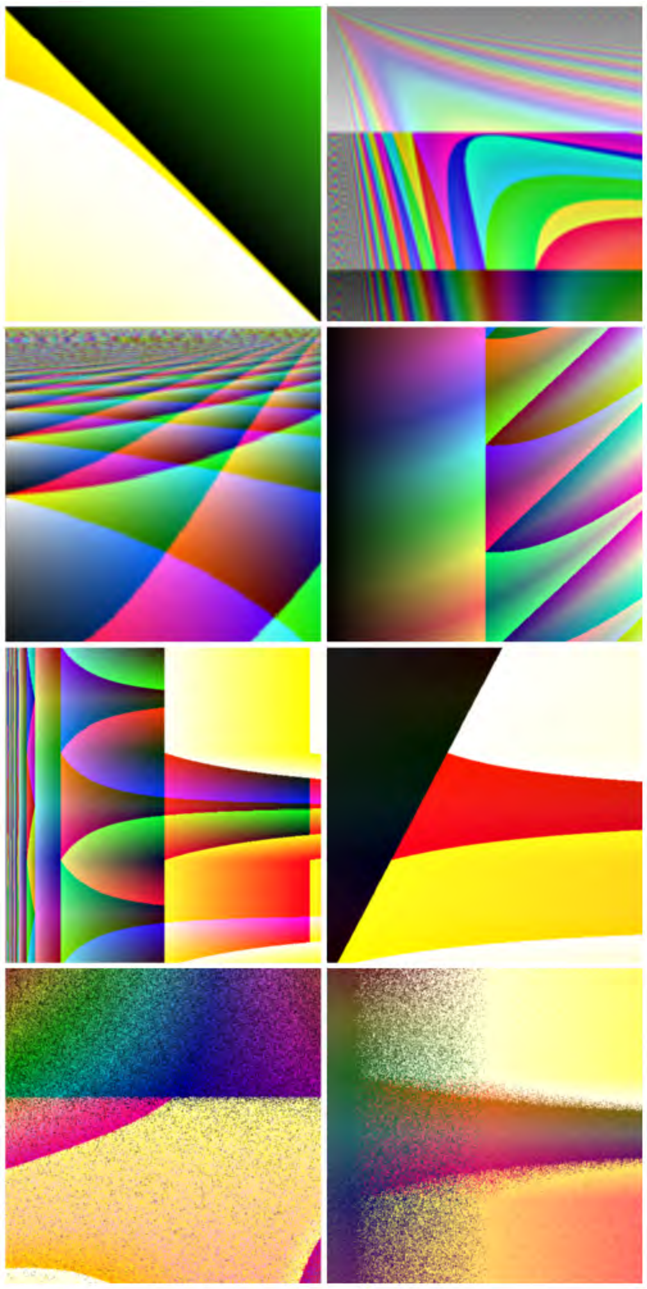


Figure 1: Evolved images - colour-based approach

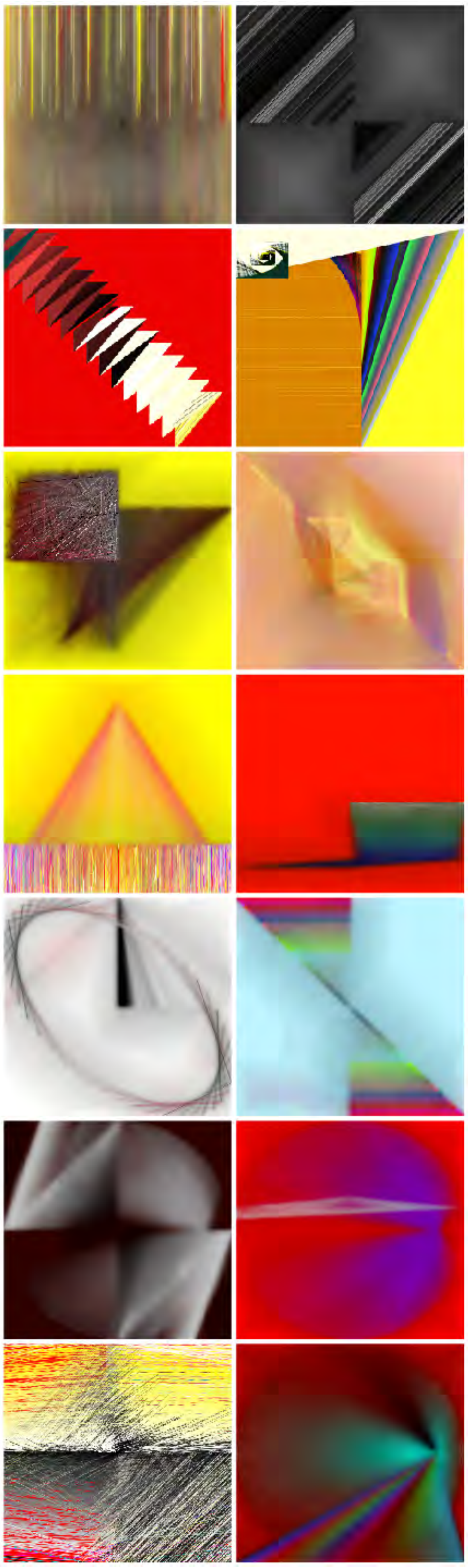


Figure 2: Evolved images - particle-based approach

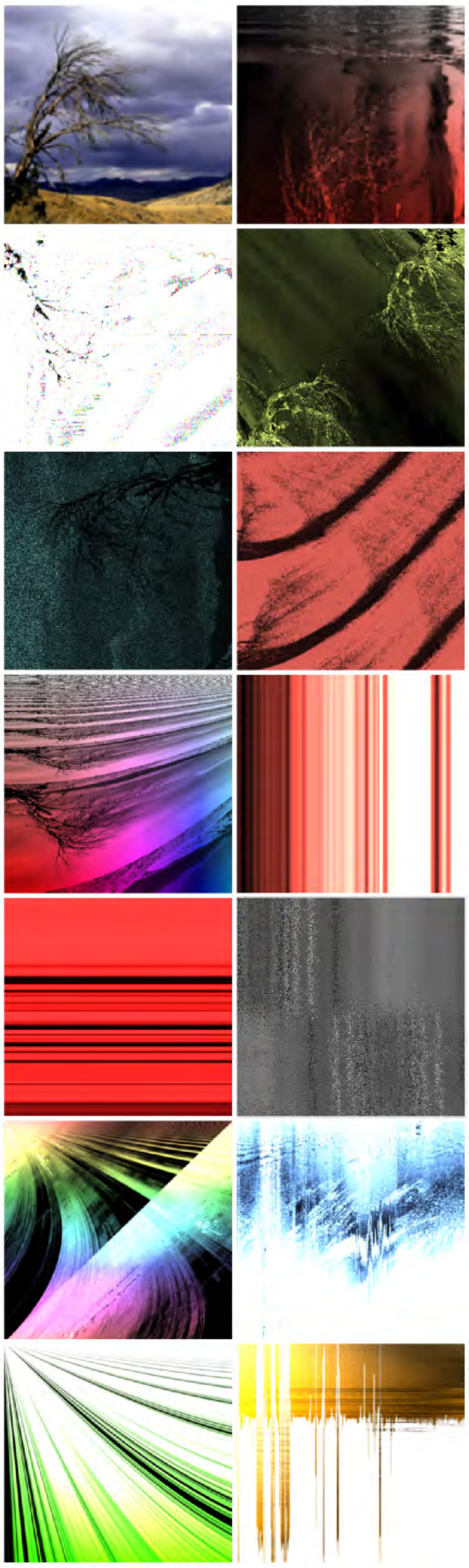


Figure 3: Original and evolved images - image-based approach

Posters

FormGrow Revisited — from DNA to 3D Organic Visualisation

William Latham
Goldsmiths, University of London
w.latham@gold.ac.uk

Miki Shaw
Goldsmiths, University of London
miki@gold.ac.uk

Stephen Todd
Frederic Fol Leymarie
Goldsmiths, University of London
New Cross, London SE14 6NW, U.K.
s.todd@gold.ac.uk
ffl@gold.ac.uk

1 Introduction

FormGrow is a "virtual machine" producing 3D computer art forms or designs. It embodies the particular "organic" aesthetics favored by Latham together with a "shape grammar" made of primitives, *e.g.*, horn-like structures, transforms or assembly rules, and a number of parameters encoding, *e.g.*, color, scale or texture. We have re-visited the *FormGrow* system of Latham and Todd (1992) and brought it back to life in a modern implementation taking advantage of standard graphics libraries and portable coding. The main emphasis here however, is on how we are bringing this system closer to the realm of biology.

Real DNA data, in the form of nucleotide sequences is transformed via a series of tables we have empirically designed to become readable by *FormGrow*. These tables process nucleotides as "codon" triplets of data as would ribosomes in a live cell. Notions of "start," "stop," and "junk" DNA code are also embedded in our system.

Our motivation for re-visiting Latham and Todd's work is that it is a powerful system which offers the possibility of generating organic-like shapes and which from its origins was meant as a metaphor to nature's way of evolving forms. In re-visiting this work, on the one hand we bring up-to-date the technology developed in (Todd and Latham, 1992) in the context of recent advances in graphics and computational geometry, and on the other hand we bring it much closer to biology via the recent advances made in understanding the working of nature in the fields of genomics and proteomics.

2 Use of DNA in *FormGrow*

DNA is alike a shape-specification language. The DNA residing in the cells of every living organism lays out its genetic blueprint. So the complete DNA, a very long string made from (4) nucleotides, of a given organism can be said to fundamentally specify its unique shape. On a lower level, DNA encodes proteins which constitute the body's key builders and building blocks. A protein is itself made of a string of (20) simpler molecules: the amino acids. The DNA translation mechanism looks at the nucleotides in groups of three: triplets called "codons;" each codon translates to a single amino acid.

Following this model, we created an analogous translation system to convert DNA sequences into *FormGrow* code. At a coarse level, *FormGrow* code can be viewed as a series of function calls, with each function requiring a small number of arguments. So we created 2 translation tables: the "transform table," which translates from codons to transformational functions; and the "number table," which translates from codons to numerical arguments. Given our input sequence, we translate the first codon into a function using the transform table, and then generate numerical arguments for that function by translating the following codons into numbers, using the number table. Once we have sufficient arguments, we return to the transform table to generate our next function, and so the cycle continues. Finally we render the generated *FormGrow* code to produce a 3D shape.

It is interesting to note some similarities between nature's translation method and ours. In the original translation table there is a "start" codon which signals that a new protein is being specified. Likewise, in our transform table, the "add horn" transform flags the beginning of a new shape. The "stop" codon is also mirrored in our system. A side effect of adopting the "start" and "stop" mechanism is that we end up with large sections of "junk code," *i.e.*, code which generates no proteins or shapes because it lies in a non-coding section of the sequence (between a "stop" and a "start"). By changing the layout of the transform table we could affect the proportion of junk code produced. We experimented with producing a few different iterations of the transform table in order to get a balance of functions that would produce an interesting variety of shapes.

At the core of this work is a simple idea of feeding DNA data sequences into a rich 3D form generator called *FormGrow*, to generate organic-looking 3D growth structure, creating an equivalence of the DNA mapped into an alternative multi-dimensional space. How useful this mapped equivalence is will become clearer as we work closer with biologists and engage in further cross-fertilization of ideas.

Reference

Todd, S. and Latham, W. (1992). *Evolutionary Art and Computers.* Academic Press.

TOWARDS CREATIVE VISUAL EXPRESSION IN VIRTUAL HUMANS

Celso de Melo **Ana Paiva**
IST-Technical University of Lisbon and INESC-ID
Avenida Prof. Cavaco Silva, Taguspark
2780-990 Porto Salvo, Portugal
cdemelo@gaips.inesc-id.pt | ana.paiva@inesc-id.pt

Virtual humans are embodied characters which inhabit virtual worlds. They introduce a new paradigm of human-machine interaction using natural multimodal communication and, thus, need to be expressive. Furthermore, expression should be effective and aesthetic. Effective means the receiver should understand the message. Aesthetic means that besides function, expression should strive for beauty. In humans, we see the former in everyday communication and, the latter, particularly in the arts.

Key to effective and aesthetic expression is a model for creativity, a model for emotions and a sophisticated medium. Given a communicative intent, the creativity model is responsible for expressing it in the medium. The emotion model affects the generation and selection of alternatives and is also central to the expression of emotions. Finally, the medium structures creative expression and, thus, should be versatile.

This work aims at creating a model for effective and aesthetic visual expression in virtual humans, Fig.1. Essentially, the communicative intent is generated in a *communicative intent planner*. The *creativity model*, then, expresses it, effectively and aesthetically, in the *bodily*, *environment* and *screen expression modules*. The *emotion model* may define communicative intent in the case of expression of emotions and may influence the creative process itself. The creativity model may also define communicative intent, a task usually referred to as problem identification, and may elicit further emotions.

Having clarified the goal, this work's contribution can now be explained. This work proposes a model for bodily, environment and screen expression as well as emotion synthesis based on the OCC emotion theory, Fig.2. These are two of the components required for effective and aesthetic expression. Furthermore, the model is fully integrated and supports sophisticated multimodal expression.

Bodily expression explores the virtual human body and face and supports: (a) keyframe animation; (b) robotics-based procedural animation; (c) psycholinguistics-based gesticulation animation; (d) pseudo-muscular facial animation. Environment expression explores the virtual human surrounding environment and supports: (a) a pixel-based lighting model which supports three types of light, multiple light sources and shadows; (b) a camera model which supports three types of cameras and a library of shots. Screen expression interprets the virtual human medium as a pixel canvas and explores: (a) filters that manipulate the scene pixels before rendering them to the backbuffer; (b) composition, where aspects of the scene are separated into layers which are filtered before combining to form the final image. Finally, emotion synthesis relies on an implementation of the OCC emotion theory with extensions to handle emotion decay, reinforcement, mood and arousal.

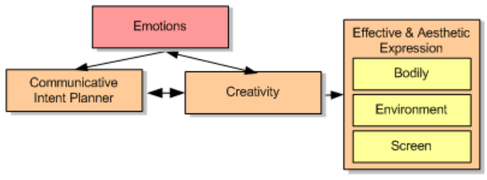

Emotions
Communicative Intent Planner
Creativity
Effective & Aesthetic Expression
Bodily
Environment
Screen

Figure 1: Overview of our approach.

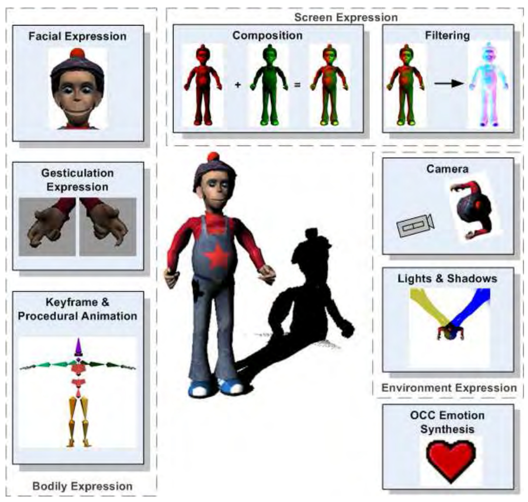


Figure 2: Overview of the model.

The proposed model is presented as a step towards effective and aesthetic virtual human visual expression. What is missing is the creativity model which converts communicative intent into bodily, environment and screen expression. To accomplish this several issues must be addressed: How does this translation occur? What aesthetic values guide this translation? How can we formalize and evaluate the aesthetics of expression? How does emotion influence the generation and selection of alternatives?