Time for AI and Society

PROCEEDINGS OF THE AISB'00 SYMPOSIUM ON CREATIVE & CULTURAL ASPECTS AND APPLICATIONS OF AI & COGNITIVE SCIENCE

17th-20th April, 2000
University of Birmingham

AISB'00 Convention

17th-20th April 2000

University of Birmingham
England

Proceedings of the AISB'00 Symposium on

Creative & Cultural Aspects and Applications of AI & Cognitive Science

Published by

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

United Kingdom

http://www.cogs.susx.ac.uk/aisb/

ISBN 1 902956 16 2

Printed at the University of Birmingham, Edgbaston, Birmingham B15 2TT, England.

Contents

The AISB'00 Convention

The millennial nature of current year, and the fact that it is also the University of Birmingham's centennial year, made it timely to have the focus of this year's Convention be the question of interactions between AI and society. These interactions include not just the benefits or drawbacks of AI for society at large, but also the less obvious but increasingly examined ways in which consideration of society can contribute to AI. The latter type of contribution is most obviously on the topic of societies of intelligent artificial (and human) agents. But another aspect is the increasing feeling in many quarters that what has traditionally been regarded as cognition of a single agent is in reality partly a social phenomenon or product.

The seven symposia that largely constitute the Convention represent various ways in which society and AI can contribute to or otherwise affect each other. The topics of the symposia are as follows: Starting from Society: The Application of Social Analogies to Computational Systems; AI Planning and Intelligent Agents; Artificial Intelligence in Bioinformatics; How to Design a Functioning Mind; Creative and Cultural Aspects of AI and Cognitive Science; Artificial Intelligence and Legal Reasoning; and Artificial Intelligence, Ethics and (Quasi-)Human Rights. The Proceedings of each symposium is a separate document, published by AISB. Lists of presenters, together with abstracts, can be found at the convention website, at http://www.cs.bham.ac.uk/~mgl/aisb/.

The symposia are complemented by four plenary invited talks from internationally eminent AI researchers: Alan Bundy ("what is a proof?"- on the sociological aspects of the notion of proof); Geoffrey Hinton ("how to train a community of stochastic generative models"); Marvin Minsky ("an architecture for a society of mind"); and Aaron Sloman ("from intelligent organisms to intelligent social systems: how evolution of meta-management supports social/ cultural advances"). The abstracts for these talks can be found at the convention website.

We would like to thank all who have helped us in the organization, development and conduct of the convention, and especially: various officials at the University of Birmingham, for their efficient help with general conference organization; the Birmingham Convention and Visitor Bureau for their ready help with accommodation arrangements, including their provision of special hotel rates for all University of Birmingham events in the current year; Sammy Snow in the School of Computer Science at the university for her secretarial and event-arranging skills; technical staff in the School for help with various arrangements; several research students for their volunteered assistance; the Centre for Educational Technology and Distance Learning at the university for hosting visits by convention delegates; the symposium authors for contributing papers; the Committee of the AISB for their suggestions and guidance; Geraint Wiggins for advice based on and material relating to AISB'99; the invited speakers for the donation of their time and effort; the symposium chairs and programme committees for their hard work and inspirational ideas; the Institue for Electrical Engineers for their sponsorship; and the Engineering and Physical Sciences Research Council for a valuable grant.

John Barnden & Mark Lee

Preface

Following on from the successful AISB'99 Convention, whose theme was the study of creativity in AI and Cognitive Science, the purpose of this symposium is to bring together researchers interested in all AI and cognitive aspects of creativity and cultural enterprise. The aim of holding one unified meeting, instead of several simultaneous smaller ones, is to promote communication between those studying different aspects of creativity, and this has been mostly achieved: the papers (and the corresponding presentations) are for the most part grouped by their relation to creativity in general, rather than to a particular domain. The exceptions to this are two papers on important low-level aspects of modelling musical creativity.

The four sections which have naturally arisen, then, are headed Exploratory Creativity, Modelling & Supporting Creative Processes, Techniques for Modelling Musical Creativity, and Methodology & Evaluation.

The first of these, in Margaret Boden's terminology, covers the modelling of creativity by traversal of a defined search space. In this volume, we see examples of mathematical and musical exploratory creativity (we see others, too, elsewhere, but presented with a different emphasis).

The second section brings together papers which focus on the modelling of particular aspects of creativity, such as metaphorical thinking. Here, we see modelling both for its own sake, and also applied to computer-assisted learning.

Section three is substantially more specific than the other sections, and includes two papers about fundamental aspects of modelling musical creativity: pattern matching in musical strings, and the exploration of pitch spaces.

Finally, and most lengthily, section four explores methodology and the question of evaluation. Creativity research currently lacks standard methodologies, and it is good to see some being developed and reported. The question of evaluation, highlighted in one sense by Margaret Boden at her AISB'99 conference keynote, actually has two senses: how can a computer program evaluate its output; and, how can we evaluate strategies for modelling creativity. Both senses are represented here, sometimes in the same paper.

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

Kim Binsted, Sony CSL, Japan
Emilios Cambouropoulos, University of Vienna, Austria
Simon Colton, University of Edinburgh, UK
Costas Iliopoulos, King's, London, UK
David Meredith, City University, London, UK
Peter Nelson, University of Edinburgh, UK
Francois Pachet, Sony CSL, France
Graeme Ritchie, University of Edinburgh, UK
Pierre-Yves Rolland, University of Paris VI, France

I am grateful also to the authors for their punctuality and cooperation in preparing this volume, especially given some difficult circumstances.

Geraint A. Wiggins
Department of Computing
City University, London
Programme Chair

I. Exploratory Creativity

Cross-domain Mathematical Concept Formation

Graham Steel, Simon Colton, Alan Bundy and Toby Walsh*

University of Edinburgh, 80 South Bridge, Edinburgh, Scotland, EH1 1HN.
*University of York, Heslington, York, England, Y010 5DD.

{grahams,simonco,bundy}@dai.ed.ac.uk, toby.walsh@cs.york.ac.uk

Abstract

Many interesting concepts in mathematics are essentially 'cross-domain' in nature, relating objects from more than one area of mathematics, e.g. prime order groups. These concepts are often vital to the formation of a mathematical theory. Often, the introduction of cross-domain concepts to an investigation seems to exercise a mathematician's creative ability. The HR program, (Colton et al., 1999), proposes new concepts in mathematics. Its original implementation was limited to working in one mathematical domain at a time, so it was unable to create cross-domain concepts. Here, we describe an extension of HR to multiple domains. Cross-domain concept formation is facilitated by generalisation of the data structures and heuristic measures employed by the program, and the implementation of a new production rule. Results achieved include generation of the concepts of prime order groups, graph nodes of maximal degree and an interesting class of graph.

1 Introduction

In previous work on automated mathematical discovery, (Lenat, 1976), (Colton et al., 1999), and in this paper, a mathematical concept is taken to mean a class of mathematical objects, such as prime numbers, square numbers, Abelian groups, complete graphs etc.[1] Mathematical concept formation is the process of identifying new classes of mathematical objects with interesting and/or desirable properties. In human mathematics, this is typically pursued in one of two ways: it can be a free exercise, in which a mathematician is looking for new things to investigate, or a more directed process, in which a mathematician is looking for a concept satisfying certain requirements as part of an investigation or a proof, see Colton (2000), chapter 3.

A mathematical domain is an area of mathematical study. Examples include number theory, the study of questions about numbers (usually meaning whole numbers), and graph theory, the study of sets of vertices, V, and edges, E, consisting of pairs of elements from E (or more informally, the study of diagrams consisting of nodes joined together with lines, see Figure 1). A *cross-domain concept* is a set of objects in one domain that are identified as a distinct class using information from at least one other domain. Some examples illustrate this idea.

The concept of *even order nodes* is cross-domain. It is the set of nodes in a graph which are joined to an even number of edges (see Figure 1). The order of a node is a graph theory concept, and the concept of even numbers is from number theory. Euler's solution to the Königsberg bridge problem, (Euler, 1736), required the use of even order nodes, and launched the field of graph theory.

The city of Königsberg in East Prussia was divided by a river containing two islands. Seven bridges connected the islands to each other and to the banks of the river, and the citizens of the city wondered if there was a way to tour the city crossing every bridge exactly once. Euler proved that this was in general possible if and only if every land mass was connected to an even number of bridges, which was not the case in Königsberg.[2]

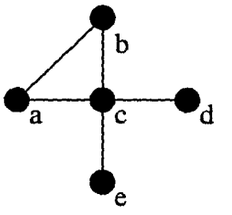


Figure 1: Illustration of even order nodes. Nodes a, b and c are even-order nodes, but nodes d and e are not

[1] A prime number is a natural number with exactly two factors, e.g. 2, 3, 5, 7. A square number is one that is equal to an integer times itself, e.g. 1,4,9,16. An Abelian group is a group in which, for all elements a, b in the group, $ab = ba$. A complete graph is one in which every node is joined to every other node.

[2] See http://www.cut-the-knot.com/do_you_know/graphs.html for more details.

The concept of *prime order groups* is another example of a cross-domain concept. A prime order group is one containing a prime number of elements. Sylow's theorem, (Sylow, 1872), required the concept of prime order groups. While not as easy to state and understand as Euler's result, it too was a breakthrough and forms perhaps the most profound result in finite group theory.[3]

Some of the most common examples of cross-domain concepts relate numbers to other domains, like the two examples above. There are also others, such as sets of matrices forming a group. A recent Fields medal winner was awarded the prize for a cross-domain investigation: Richard Borcherds proved the 'Moonshine Conjecture', originally proposed by John Conway and others, (Conway and Norton, 1979). This theorem links elliptic modular functions with concepts from string theory and group theory.[4]

Cross-domain reasoning also arises in the natural sciences, most commonly when some mathematics is applied to experimental results. For example, Mendel founded the field of genetics when he applied some elementary combinatorics to the results of his famous pea experiments.

In summary, we've seen that cross-domain concepts, while not dense in the mathematical literature, often provide the inspirational step leading to results of real importance. These ideas represent what we think of as creative stages in the development of a theory. An effective automated mathematical discovery package should have the ability to form cross-domain concepts, and therefore be able to provide the inspiration for such creative steps.

1.1 Background

The first automated mathematical concept formation program was Lenat's *AM*, as described in Lenat (1976). *AM* stored concepts as definitions in LISP, and modified old definitions to create new concepts. It also made conjectures about the concepts it had invented. Lenat provided *AM* with 76 initial concepts, mostly referring to bags, sets and set operations. When running, *AM* developed set-theory based definitions of numbers. It then quickly moved into number theory, and re-invented some famous concepts such as highly composite numbers. Lenat followed up *AM* with *EURISKO*, Lenat (1983), which was given some number-theory and set-theory concepts to start with. However, neither program had facilities for explicitly reasoning about the two different domains in order to propose cross-domain concepts.

[3]For those interested, Sylow's three-part theorem built on a result of Cauchy, which stated that a group whose order is divisible by a prime p has an element of order p. Sylow extended this to the following: If p^n is the largest power of the prime p to divide the order of a group G then (i) G has subgroups of order p^n (ii) G has $1 + kp$ such subgroups for some k, (iii) any two such subgroups are conjugate. Almost all work on finite groups uses Sylow's theorems. (O'Connor and Robertson, 1999)

[4]There is a good, short summary of the story of the moonshine conjecture at http://www.sciam.com/1998/1198issue/1198profile.html

GT[5], Epstein (1988), was written by Epstein to perform concept formation, conjecture making and theorem proving in graph theory. In GT, a graph type is represented by a seed, S, consisting of a set of base cases for the type, a constructor, f, and a set of constraints for the constructor, σ. These last two together describe a correct and complete construction of all graphs in the class. GT formed new concepts by generalising, specialising and merging old ones, performing heuristic search on a best-first agenda basis. At least one of the conjectures proposed and proved by GT involved cross-domain concepts:

> There are no odd-regular graphs on an odd number of vertices.

This theorem states that, given an odd number of points, there is no way to join them up such that every point is connected to the same *odd* number of other points. However, no other cross-domain conjectures are reported. GT eventually succumbed to the size of the search space and stopped producing concepts and conjectures that the program's author considered interesting.

The HR[6] program, Colton et al. (1999), is an automated concept formation which also makes conjectures about its concepts and calls on theorem proving and model generation tools to settle them. At the start of an HR session, a domain is chosen, some initial objects in that domain are provided, and axioms for that domain are specified by the user. All model generation and theorem proving tasks are then carried out with respect to these axioms. HR will then proceed to form concepts within that domain, using general production rules such as 'conjunct' (pick out objects satisfying the definition of two previous concepts) and 'common' (pick out pairs of objects sharing a particular property) to make new concepts. A weighted sum is taken of a variety of heuristic measures to evaluate the interestingness of a concept. HR will then apply more production rules to the most interesting concepts. However, the original version of HR could not store objects from more than one domain at a time, so could not reason about multiple domains in order to form cross-domain concepts. The HR project is ongoing, and a new version is being developed in Java, (Colton et al., 2000).

1.2 Paper outline

In the rest of this paper, we describe the generalisation and extension of the HR program to perform cross-domain concept formation. In section 2, we describe the operation and knowledge structures of the HR program, and how they were altered to allow cross-domain concept formation. In section 3 we discuss the results achieved by the new version of the program looking for cross-domain

[5]GT stands for Graph Theorist

[6]HR is named after Hardy and Ramanujam, two famous mathematicians who worked together for a period early in the twentieth century.

Table 1: Concept table for 'group operation' (rows for 2 groups shown)

Group	Element	Element	Element
c2	e	e	e
c2	e	a	a
c2	a	e	a
c2	a	a	e
c3	e	e	e
c3	e	a	a
c3	e	b	b
c3	a	e	a
c3	a	a	b
c3	a	b	e
c3	b	e	b
c3	b	a	e
c3	b	b	a

concepts in graph theory and number theory. These included rediscovery of nodes of maximal degree, discovery of an interesting type of graph and a generally high yield of good quality concepts. In section 4, we draw some conclusions and outline some suggestions for further work.

2 Concept formation in HR

There follows a brief description of the HR program. Further details can be found in (Bundy et al., 1998) or on the HR project webpage[7].

2.1 Representation

Designed initially to work with finite algebras, HR can work in any finite mathematical domain. The user must supply a set of axioms for the domain, and a set of initial concepts to work with. Typically, these will consist of a set of entities from the domain, together with a way of breaking them down. So, for example, in group theory the user might supply the Cayley table[8] for some small groups, or in number theory, the breakdown of the first 10 integers into their divisors.

In HR, each concept is represented as a data table, and each entity to which that concept is applicable will have one or more rows in the table. The first column in the table identifies the entity, and the other columns refer to properties of the entity or its components. For an example, see table 1, the Cayley table for two cyclic groups.

2.2 Operation

HR functions on a best-first heuristic search basis. New concepts are generated by HR's 9 production rules, and evaluated using 7 heuristics. A weighted sum of the seven scores is calculated for each concept, and the concepts are sorted into order, the highest scoring concepts coming first. The weights for each heuristic are set by the user.

2.3 HR's heuristics and production rules

HR uses the following heuristics to measure the interestingness of a concept:

Parsimony Concepts with small data tables tend to describe objects more parsimoniously, so these are scored positively.

Discrimination and Invariance The user can supply a 'gold standard' categorisation, and these two heuristics will measure how close a concept comes to achieving that categorisation.

Complexity This is a measure of how many production rule steps were applied to build the concept. Simpler concepts are preferred.

Applicability The proportion of entities known to HR which are referred to by the concept is the applicability.

Novelty If the categorisation produced by a concept hasn't been seen before, then it is a novel concept, and so scores highly. Categorisations that have been seen many times before yield lower novelty scores.

Provable facts Concepts relating to proved theorems are scored highly.

The production rules used to form new concepts are:

Size Measures the size of a set, e.g. given a table of integers and their divisors, it will produce a table showing how many divisors each number has.

Split Picks out an integer value - usually 1 or 2. Given a table showing how many divisors a number has, the split rule can pick out integers with exactly 2 divisors (prime numbers).

Match Produces a table containing rows with matching columns. So, given a table of integers decomposed into their factors, it can pick out numbers which decompose into two identical factors, i.e. square numbers.

Forall Finds a set of entities, all of which have sub-objects of a certain type. Given the concept of central elements in a group, it will produce Abelian groups (i.e. groups in which all elements are central).

Conjunct Forms the conjunction of two previous concepts, e.g. from square numbers and prime divisors, it can form squares of primes.

[7] http://www.dai.ed.ac.uk/~simonco/research/hr

[8] A Cayley table for a group gives the result of applying the group's multiplication operation to any pair of elements.

Exists Removes a column from a data table. So, given a table consisting of groups and their elements of order 3, it will produce a table of all groups which contain an element of order 3.

Compose Given two concepts representing a function, this production rule will produce a data table representing their composition. For example, given a data table containing groups and their sizes, and a data table containing integers and their prime factors, the compose rule would produce a table containing groups and the prime factors of their size.

Common Picks out entities with rows in common, e.g. integers sharing the same prime divisors.

Negate Returns the entities and corresponding rows *not* appearing in a previous concept's table with respect to the complete list of entities provided by the user. For example, given the concept of even numbers, this production rule will produce the concept of odd numbers.

Note that none of the production rules are domain specific, and all perform only very basic operations.

2.4 Adapting HR for Cross-domain Work

We now look at the modifications made to HR in order to allow it to form cross-domain concepts. As we mentioned above, Colton's HR program performs not only concept formation but also conjecture making, theorem proving and counterexample finding. In the work reported here, however, only the concept formation capabilities of the program were used. This was because HR currently lacks the ability to interface with inductive theorem provers, and so would be unlikely to be able to prove any conjectures involving numbers or graphs. It was anticipated that all the testing would involve number theory, so the theorem proving mechanism of the program was switched off for this work.

The first stage of the development work involved making some changes to the way HR stores concepts in order to be able to distinguish between concepts from different domains. HR stores a list of 'entities', e.g. a list of groups or a list of numbers depending on the working domain. This was changed to allow entities to be identified with their domains.

Some small changes were required to the way HR builds new concepts. For example, `negate`, returns a table containing entities not satisfying a previous concept definition. This was changed to test the domain of the previous concept, and return a table containing only those entities *from the same domain* which do not satisfy the previous concept definition.

Certain aspects of the way HR measures the interestingness of its concepts also required modification. One heuristic HR employs measures the 'applicability' of a concept, i.e. the proportion of entities to which it applies. This had to be changed so that HR only measured how many entities in the concept's own domain were referred to by the concept. The 'complexity' measure was also changed so as to be more lenient towards cross-domain concepts. The measures concerned with classifications (discrimination, invariance and novelty) required changes to allow meaningful measurements to be made with respect to the applicable domain. These changes consisted of directing HR to only refer to categorisations in the same domain when making the measurements.

2.4.1 User settings to control cross-domain work

Some new settings were added to allow the user to control the multi-domain aspects of an investigation. The first alteration was to set up a system whereby the user can prescribe the amount of investigation to take place in each domain presented. This was achieved by adding user settable options `domain_list` and `domain_multiplier`. The user sets `domain_list` to be a list of domains, say `[group,integer]`, and `domain_multiplier` to be a number, say 50, and then HR will produce 50 concepts in graph theory, 50 concepts in number theory, and then carry on pursuing whatever it finds most interesting. This setting has no effect on the cross-domain aspect of the investigation, it simply specifies the domain that a concept should refer to.

The second change was to add two more user options to control the cross-domain behaviour of the program, `no_cross_before` and `encourage_cross_after`. The user sets `no_cross_before` to a value, say 150, and then no cross-domain concepts will be formed until 150 single-domain concepts have been formed. `encourage_cross_after` is set to a value, say 200, and then after 200 concepts have been formed, cross-domain pairs will be chosen first by the two-table production rules.

A further change was to relax the complexity limits for cross domain concepts. HR has a user settable option called `complex_max` which sets a depth limit for the search. This was modified so that a concept relating objects from n domains could be built on up to a depth of n times the complexity limit set.

2.4.2 A new production rule: `extreme`

Although HR's existing multi-concept production rules could be made to function across multiple domains (with minor modifications), it was important to establish the efficacy or otherwise of production rules designed explicitly to facilitate cross-domain reasoning. We decided to develop a rule to introduce extremes of orderings into HR. Several previously inaccessible concepts require the use of an ordering. For example, in graph theory we may be interested in the node of maximum degree, the largest clique, or the longest path. In group theory, we might be interested in the elements of maximal order or the largest proper subgroup. In number theory, we might be interested in the largest prime divisor or largest common

Table 2: extreme with parameters (1,3). Bold rows are extracted

Group	Element	Number
G0	a	1
G0	b	2
G0	**c**	**3**
G1	a	1
G1	**b**	**2**

Table 3: extreme with parameters (1,3). Bold rows are extracted

Graph	Node	Number
G0	a	4
G0	a	5
G0	a	7
G0	b	1
G0	**b**	**9**

factor. We needed to keep the rule general, to allow HR to use any ordering it has available, and to allow it to orient an ordering in either direction, in order to be able to extract both maximal and minimal values.

This new production rule was called extreme. It takes in two pre-existing concepts, and treats one of them as an ordering. It then takes the other concept and extracts only those rows whose entry in a specified column (first parameter) are the 'largest' for a specified entity (second parameter: graph, node, group member, etc.) with respect to the ordering chosen. If several rows share the same extremal value, they are all extracted. Tables 2, 3 and 4 show examples, using the ordering in table 5.

Note that no ordering properties such as asymmetry or transitivity are assumed or checked for when choosing a concept to use as an ordering for the extreme rule. This was a deliberate decision, taken for several reasons: firstly, checking that a table has the necessary properties to qualify as a partial or total ordering would be computationally expensive. Secondly, it was anticipated that completely inappropriate ordering tables would tend to give empty tables, i.e. would not have any extremal values, and so HR would not form a new concept (it would instead conjecture that such a value did not exist). This was borne out by the results achieved. Thirdly by keeping the rule as general as possible we were giving HR a chance to come up with something truly novel using a table as an 'ordering' that a human mathematician had never previously considered using. The original HR project had followed a pro-generality methodology, and we wanted to preserve this in our extensions.

Table 4: extreme with parameters (2,3). Bold rows are extracted

Graph	Node	Number
G0	a	4
G0	a	5
G0	**a**	**7**
G0	b	1
G0	**b**	**9**

Table 5: Ordering - this concept is given to HR when working in number theory

Number	Number
1	0
2	0
2	1
3	0
3	1
3	2
⋮	⋮

3 Results

We evaluated the final product with respect to several precise criteria, in the hope that all these measures together would give an accurate account of the degree of success of the project. The two main areas of testing were: an evaluation of HR's ability to spot classically interesting cross-domain concepts and evaluation of the quality of the new concepts output by HR. There follows a description of the testing criteria, the methods used for testing and the results. In this paper, for considerations of space and intelligibility to non-mathematicians, we report only the results obtained in graph theory and number theory. Details of group theory and number theory testing, and further details of the graph theory testing including the complete output from a run, can be found in (Steel, 1999).

3.1 Generating standard interesting concepts

To measure the ability to re-invent standard cross-domain concepts, we compiled a list of target concepts for which the original HR implementation was capable of finding the individual single domain concepts required. Then we ran HR in the two domains, and examined the output to see how many of these concepts had been rediscovered. The methodology used was the following: first set HR up with the correct production rules to build the single domain concepts that are required to form the cross-domain target concepts. Then, set the weights for the concept measuring heuristics, and set HR off to form 500 concepts.

Several 500 concept batches were run with different

Table 6: Graph theory and number theory target concepts. The double line separates 'core' concepts from 'peripheral' concepts

Concept	Reason for Inclusion
Eulerian graphs	A result in the first graph theory paper, Euler (1736).
Maximal order nodes	Used in many inequalities of invariants.
Minimal order nodes	Used in the greedy graph factorization algorithm and several inequalities.
Odd order nodes No. of odd order nodes	A graph contains an Eulerian path if it has 0 or 2 odd order nodes.
Even order nodes	Needed for Eulerian graphs.
Number of nodes of max/min order	Give an indication of the structure of the graph.
Order of a star graph centre	Identifies star graph uniquely.
Star graph with an even order node	Characterises symmetric star graphs.

weights assigned to the concept measures. After each run, weights were altered to try to favour the root single domain concepts required to build the missing targets. The idea behind this was that if the target concepts really were representative of the kind of concepts we want HR to find, then by adjusting the weights to perform well on these concepts, we would not be 'over-tweaking' but rather determining a set of weights well-suited to concept formation in that particular pair of domains. In the event, after four 500 concept runs, we had found a set of weights that gave our best performance on the target set.

3.1.1 What is a 'classically interesting' concept?

Cross-domain concepts, whilst vital to mathematics, are not particularly dense in the literature. This is inconvenient, as we need a significant number of target concepts to get representative test results. Consequentially, a small number of vital concepts were picked out and identified as 'core' target concepts. Then, some slightly more obscure concepts that could still be considered interesting were identified, and labelled as 'peripheral' target concepts. Table 6, gives the test concepts for the runs in graphs and numbers, and the reasons for their inclusion.

3.1.2 Standard concepts reinvented

HR was able to reinvent 2 out of 3 of the core targets and 3 out of 5 of the peripheral targets in the testing undertaken. The missed targets were traced to a particular step in the investigation where HR applied its `exists` production rule in such a way as to miss abstracting away the actual order of the nodes. This can be easily fixed - a version of HR is under development that will insist on applying a production rule with all possible parameters before attention is switched to another concept, or another rule.

3.2 Inventing new concepts

We measured the interestingness level of all the concepts produced in a 500 concept run on the following scale:

Type 1 - Concepts in the classical target set (core or peripheral).

Type 2 - Concepts of a similar level of interest to those in the peripheral target set, of interest but only in specialised areas of the theory. For example, non-regular graphs, graphs with all nodes of order greater than one and nodes of prime order were all found in one test run and classified as type 2.

Type 3 - Concepts which may be of interest, but only in a specialised situation. For example, graphs with more than two nodes of order 1 and graphs where all nodes are of order either a or b, and $a + 1 = b$.

Type 4 - Anything not falling into the above three types

To analyse thoughtfully and accurately the quality of cross-domain concepts in a 500 concept run with respect to our four-point scale was a time consuming process, and only one run was completely analysed in this way. The run chosen was the run in graph theory and number theory that produced the most classical target concepts. A complete list of the concepts evaluated, and their classifications, is given in appendix 1 of (Steel, 1999).

HR's own measures of interestingness were not referred to at this stage of the evaluation process. By default, HR re-evaluates all the concepts it has invented so far every time it invents another 10 new concepts. After this re-evaluation, the top ten concepts (ranked by interestingness) are displayed on the screen. For this evaluation run, the program was altered to prevent it from displaying the top ten list, to ensure that the concepts were categorised purely on their apparent mathematical merit. This allowed us to compare HR's measure of interestingness with our own later (see section 3.3).

The proportion of concepts fitting into each of the four categories are illustrated in the pie chart in figure 2. We can classify a concept satisfying the criteria for a type 1, type 2 or type 3 classification as being 'acceptable', in that it must at the very least be plausible and of some interest. Of the concepts in the run analysed, 56% were acceptable. This compares favourably with Lenat's 125 out of 300 (42%)'acceptable' concepts in *AM*, and 200 out of 1000 (20%) for *EURISKO*. Few very interesting concepts (i.e. type 1 or type 2) were found in the run (just 8% of the total cross-domain concepts). This is comparable with Lenat's 25 out of 300 'really interesting' concepts in *AM* (8.3%). Of course, we must be wary of attaching excessive importance to subjective judgements of this kind without knowing exactly what Lenat was classing as an 'acceptable' or a 'really interesting' concept.

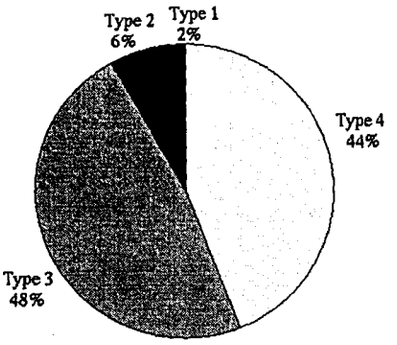


Figure 2: Overall concept quality

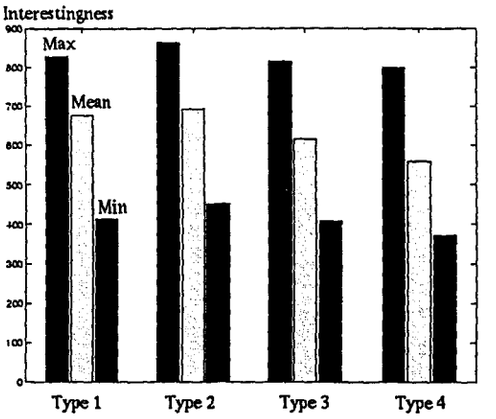


Figure 3: HR's evaluation of concepts' interestingness

3.3 Effectiveness of HR's interestingness measures

An effective concept formation program should rate as interesting the same concepts a human mathematician would rate as interesting. We evaluated this by collating the interestingness measures assigned to the concepts in the fully evaluated run presented in figure 2. The maximal, mean and minimal interestingness scores for each category of concept are illustrated in bar chart form in figure 3. There is a significant fall in the mean level of interestingness (the central, light coloured bar in the chart) from type 2 to type 3 to type 4, but a smaller fall in the maximal and minimal values. The spread of interestingness values assigned between maximum and minimum for any given concept type was more significant than the scale of the decrease in the mean value. This is unfortunate, as it means HR can be misled: it could assess a type 4 concept to be more interesting that a type 1 concept. HR cannot currently undertake any mathematical investigation of these concepts as its proof tools are limited to first order logic. HR has no background knowledge of the mathematical literature. Despite this, we are asking it to give an instant evaluation of the mathematical worth of a concept. Given the scale of this task, achieving a general correlation with the human assessment of interestingness is a good result, but the degree of mis-assessment imposes a limit on the success of the program. Some suggestions for improvements are given in section 4.

3.4 Discovery highlights

A good automated concept formation program should come up with some new novel and interesting concepts. To evaluate this, one interesting looking concept was extracted from the analysed graph theory run. It had the following definition:

> A Graph G with a node $n1$ of order M such that $\forall$ nodes $n2 \in G$, order$(n1) \geq$ order $(n2)$, and $M = \{I | \exists$ a node $n3 \in G$,order$(n3) = I\}$.

This corresponds to a graph with a node of order M that is the maximal order node in a graph in which there are nodes of M different orders. The definition is fairly complicated, but a moment's thought reveals that such a graph must have at least one node of every order from 1 to M. What is the minimum number of nodes that a simple graph[9] with such a property can be drawn on for a particular M ? We managed to prove the following simple theorem:

Theorem 1 $\forall m \in \mathbb{N}, m \geq 1, \exists G$, *a graph on* $m+1$ *vertices s.t.* $\forall n, 1 \leq n \leq m, \exists$ *a node of order* n *in* G

Proof The proof is by induction on m, details in (Steel, 1999).

This type of graph was new to the authors, but it has appeared in mathematical literature. In (Zeitz, 1999), a problem is posed involving a host inviting 10 couples for a party:

> I ask everyone present, including my wife, how many people they shook hands with. It turns out that everyone shook hands with a different number of people. If we assume that no one shook hands with his or her partner, how many people did my wife shake hands with? (I did not ask myself any questions.)

[9] A simple graph is one with no duplicated connections and no loops.

By drawing a graph, which turns out to be of the type we rediscovered, and applying a little induction, Zeitz shows that the hostess must have shaken 10 hands.

So, our graph theory concept is involved in at least two simple but interesting pieces of mathematics. This is a promising result. The concept was also a complete surprise to the authors - it seems to be a characteristic of HR that it is able to find a way of inventing concepts which at first thought one would not expect it to have the capacity to represent. HR finds them interesting because it can evaluate qualities of the underlying data rather than just the definition.

4 Conclusions and Further Work

The control structure of the cross-domain HR could be modified so that cross-domain concepts are only formed when they are needed to develop a theory further. HR could have a pre-set interestingness limit, and attempt to generate cross-domain concepts only when formation in a single domain was producing concepts below that threshold setting.

As highlighted by the bar chart in figure 3, there is room for improvement in HR's interestingness heuristics. One easy way to improve performance would be to increase the number of models HR is given in each domain. This would make the applicability measure more accurate, and so decrease the interestingness of concepts which consist of a convoluted definition of one particular model. However, it would also slow the program down.

The conjecture making and theorem proving aspects of the HR project have not been extended in this project. Allowing HR access to inductive theorem provers would give it a chance to prove some cross-domain conjectures involving numbers. In particular, many graph theory proofs are based on induction. Being able to prove cross-domain conjectures would also allow HR to judge the worth of cross-domain concepts more accurately, as it does in the single domain version.

Designing and adding further production rules would enhance HR's coverage of mathematics. A useful exercise would be to pick some concepts, say from an index or glossary in an undergraduate text, and analyse what kind of rules would be required to build those concepts. This would at least give a feel for the scale of the problem, i.e. whether we need ten more production rules or a thousand more production rules. It would be useful to carry out this kind of analysis before embarking on further extensions to the work.

The results achieved by the program were generally encouraging. In particular, the ideas behind HR were seen to generalise to a search space with a much larger branching factor without destroying the quality of the concepts constructed. If the cross-domain conjecture making abilities of HR can be extended similarly, and there is every reason to believe that they can, then perhaps a future version of HR will be able to come up with a discovery of real mathematical importance.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments. The work reported here was supported by an EPSRC grant.

References

A. Bundy, S. Colton, and T. Walsh. HR - automatic concept formation in finite algebras. Technical Report 920, (Presented at the Machine Discovery Workshop at ECAI 98) Department of Artificial Intelligence, University of Edinburgh, 1998.

S. Colton. *Automated Theory Formation in Pure Mathematics*. PhD thesis, Division of Informatics, University of Edinburgh, 2000.

S. Colton, A. Bundy, and T. Walsh. HR: Automatic concept formation in pure mathematics. In *IJCAI '99*, 1999.

S. Colton, A. Bundy, and T. Walsh. Multi agent theory formation in mathematics. In *Creative and Cultural Aspects of AI*, Birmingham, England, 2000. AISB.

J. Conway and S. Norton. Monstrous moonshine. *Bull. Lond. Math. Soc.*, 11:308 – 339, 1979.

S. Epstein. Learning and discovery: One system's search for mathematical knowledge. *Computational Intelligence*, 4(1):42–53, 1988.

L. Euler. Solutio problematis geometrian situs perinentis. *Opera Omnia*, 7(1):128–140, 1736.

D. Lenat. *AM: An artificial intelligence approach to discovery in mathematics*. PhD thesis, Stanford University, 1976.

D. Lenat. Eurisko: A program which learns new heuristics and domain concepts. *Artificial Intelligence*, 21: 61–98, 1983. The Nature of Heuristics III.

J. O'Connor and E. Robertson. The mactutor history of mathematics. Available on the web at http://www-groups.dcs.st-and.ac.uk/history/index.html, 1999.

G. Steel. Cross-domain concept formation using HR. Master's thesis, University of Edinburgh, 1999.

L. Sylow. Théorèmes sur les groupes de substitutions. *Mathematische Annalen*, 5:584–594, 1872.

P. Zeitz. *The Art and Craft of Problem Solving*, chapter 3, pages 84–85. Wiley, 1999.

Agent Based Cooperative Theory Formation in Pure Mathematics

Simon Colton, Alan Bundy and Toby Walsh*

University of Edinburgh, 80 South Bridge, Edinburgh EH1 1HN, United Kingdom.
*University of York, Heslington, York Y010 5DD, United Kingdom.

simonco@dai.ed.ac.uk, bundy@dai.ed.ac.uk, toby.walsh@cs.york.ac.uk

Abstract

The HR program, Colton et al. (1999), performs theory formation in domains of pure mathematics. Given only minimal information about a domain, it invents concepts, make conjectures, proves theorems and finds counterexamples to false conjectures. We present here a multi-agent version of HR which may provide a model for how individual mathematicians perform separate investigations but communicate their results to the mathematical community, learning from others as they do. We detail the exhaustive categorisation problem to which we have applied a multi-agent approach.

1 Introduction

Automated theory formation in pure mathematics involves the production of mathematical concepts, examples, conjectures, theorems and proofs. Various systems have modelled different aspects of theory formation. The AM program, Davis and Lenat (1982), worked in elementary number theory and modelled how an exploratory approach can drive theory formation. It used heuristics to guide the search towards more interesting concepts and achieved some success re-inventing well known concepts. The GT program, Epstein (1988) worked in graph theory and was the first to model the use of theorem proving to help direct theory formation. The IL program, Sims and Bresina (1989), constructed operators with given properties over types of numbers such as complex numbers. This modelled how theory formation can be goal directed.

The AM program stopped being productive after a while in every session. AM's author, Lenat, argued that this was because it needed more heuristics and the ability to invent its own heuristics. He implemented the Eurisko program to do this, Lenat (1983), but it but wasn't as successful as AM and it is debatable whether it added to our understanding of theory formation. In Furse (1990) several other reasons are given why AM 'ran out of steam'. One reason based on arguments from Kuhn (1970) is that AM does not model any social aspect of the mathematical community. That is, creativity in mathematical research often arises from the interaction of several mathematicians, often collaborating on the same problem but sometimes working on different problems, possibly even in different domains.

To our knowledge, no theory formation program in mathematics has modelled the communication of ideas between mathematicians. We have extended the HR program, Colton et al. (1999), to model limited interaction between different copies of the program running concurrently. This multi-agent approach has led to greater creativity in the system as a whole. In §2 we give necessary background information about the HR system, followed in §3 by details of the multi-agent implementation. In §4 we discuss the exhaustive categorisation problem to which we have successfully applied multi-agent theory formation. In §5 we discuss further possibilities for this approach, including an application to the problem of integer sequence extrapolation.

2 The HR System

The HR system models the major activities of mathematical theory formation: forming concepts, calculating examples, making conjectures, proving theorems and finding counterexamples. The version of HR discussed in Colton et al. (1999) and Steel et al. (2000) is a Prolog implementation which includes all of this functionality. Java is a more natural language for implementing agent based programs and the version of HR we discuss here is a re-implementation of HR in Java which is still under development. The Java version does not yet have conjecture making or theorem proving abilities, so theory formation is limited here to the compilation of concepts. HR does this by exploring a space of concepts using a best first search based on measures of interestingness.

The user supplies a set of **objects of interest** for the domain, eg. the numbers 1 to 10 in number theory. They also supply a set of initial concepts by providing a **definition** in terms of a set of predicates, the conjunction of which defines the concept, and an exhaustive **datatable** of examples calculated for all the objects of interest. For instance, the concept of multiplication in figure 1 is supplied in number theory with a datatable of examples where the first column contains integers, which the integers in the second and third columns multiply to give. A definition is also supplied for multiplication as a set of six pre-

dicates describing the triples $[n, a, b]$ in the datatable:

(i) n is an integer (ii) a is an integer (iii) b is an integer
(iv) a divides n (v) b divides n (vi) $a \times b = n$

Theory formation proceeds in **theory formation steps**: HR takes a concept already in the theory and passes it through a **production rule** (detailed below) along with a parameterisation detailing exactly what the production rule should do. The production rule will generate the definition and datatable of a new concept. HR then checks whether it has a concept in the theory already with exactly the same datatable. If it finds a match, the definition for the new concept is added as an alternative definition to the old concept, and the new concept is discarded. In fact, if the new concept is less complex (as defined below) than the original, HR replaces the original concept with the simpler new one. If the new concept does not match one already in the theory, it is added to the theory.

Currently HR has uses just 7 production rules. The types of concepts they produce is described briefly below. It is perhaps easiest to imagine that the objects discussed are integers and the subobjects are divisors.

- The **exists** rule produces concepts identifying objects where there exists a subobject of a particular nature.
- The **forall** rule produces concepts identifying objects where all subobjects are of a particular nature.
- The **size** rule produces functions which count the number of subobjects of a particular nature for each object.
- The **split** rule produces concepts identifying objects with a particular number of subobjects.
- The **match** rule produces concepts identifying objects with equal subobjects of a particular nature.
- The **negate** rule produces concepts identifying objects which have the properties described by one old concept but not the properties of another old concept.
- The **compose** rule produces concepts identifying objects which have the properties of two old concepts.

Note that the first five rules are called **unary** production rules as they produce a new concept from only one previous concept. The last two are called **binary** production rules as they produce a new concept from two previous ones. In figure 1 we see that to construct the concept of square numbers from the concept of multiplication, two theory formation steps are required. Firstly, the match production rule is used to construct the concept of integers and their *integer* square roots. Secondly, the exists production rule is used to identify those integers for which there exists such an integer square root, namely 1 and 4. For a more detailed description of the production rules, please see Colton et al. (2000a).

The **construction history** of a concept is the set of triples of (old concept, production rule, parameterisation) which detail the steps used to build all the previous concepts upon which the concept is based. Given the construction history of a concept and the user supplied concepts upon which it is ultimately based, HR can completely re-construct the concept by following the steps in order. We say the **complexity** of a concept is the size of its construction history. Even with just seven production rules, the size of the space HR searches when forming a theory is very large. To limit the search, we usually employ a complexity limit of between 5 and 10, ie. no theory formation steps involving concepts with complexity greater than the limit are allowed.

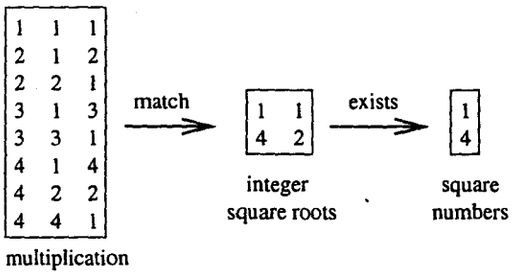


Figure 1: The construction of square numbers

Each new concept formed is added to the **agenda**. If a concept reaches the top of the agenda, all theory formation steps involving it are carried out until it is replaced at the top. HR can perform a **breadth first search** which puts every new concept to the bottom of the agenda, and a **depth first search**, which puts them at the top. Alternatively, HR can perform a **unary first** search which combines the breadth first and depth first searches. In a unary first search, HR uses the unary production rules in a depth first manner, but the binary production rules in a breadth first manner. This means that any new concept introduced is explored thoroughly with the unary rules, but not combined with other concepts until much later.

To enable effective traversal of the space we have enabled HR to employ a **best first search**: after every step it determines which is the most interesting concept and moves this to the top of the agenda. HR has many different measures available to estimate the interestingness of a concept, as detailed in Colton et al. (2000b), and uses a weighted sum of values calculated for a particular concept to determine the overall worth of the concept.

We discuss only the **novelty** measure here. To define this, we note that we can use the examples of a concept to produce a **categorisation** of the objects of interest in the theory. For instance, the square number concept in figure 1 categorises integers 1 to 4 as: $[1, 4], [2, 3]$ because 1 and 4 are squares, 2 and 3 are not. For every concept in a theory, HR can determine the categorisation it produces for the examples supplied by the user. The categorisation for a particular concept may not be unique and we define the novelty of a concept to be the reciprocal of the number of other concepts which share its categorisation. Therefore, concepts producing categorisations unique to them score 1 as they are novel, but concepts producing categorisations which also belong to 99 other concepts score $1/100 = 0.01$ as they are not novel.

3 Multi-Agent HR

Given a problem to solve, one approach is to employ a set of autonomous programs, called **agents**, each with different abilities and tasks and each able to communicate with the others. The set of agents forms an **agency** and it is hoped that dividing tasks between the agents will improve the overall efficiency of the system. Often each agent runs on a separate processor, and improvements in efficiency are observed as a result of the parallel attack on the problem. For our purposes, using an agency allows us to model a community of mathematicians each performing individual investigations but communicating their results to others.

Our implementation of an agency is fairly straightforward. Using the multi-threading capabilities in Java, we run several copies of HR as individual threads. Therefore our agency runs on the same processor, not on parallel processors, although this could easily be altered to improve efficiency. Each agent has a different name to identify the concepts they introduce, and each has different settings which guide its theory formation. We also run a 'watcher' program in another thread which determines when the task set for the agency has been achieved, and stops the agents when this is the case.

Communication between agents is limited to sending and receiving concepts. Each agent has a set of **inboxes** into which they receive concepts from the other agents, with a different inbox for each other agent. There is no global repository to which concepts are sent and taken, and the user can customise each agent to control which concepts from which inboxes it takes. It is hoped that the communication of a concept will increase the number of ways it is developed. For example, the binary rules combine two concepts. As the concepts available to each agent will be different, a concept read by one agent will be developed differently to the way in which it will be developed by the agent which sent it.

Each concept resides as an object of class Concept in the Java program, and each object contains all relevant information about the concept, including the information representing the concept and the values calculated to assess it. To communicate a concept, a pointer to the concept is put in the inboxes of all the other agents. Therefore, the receiving agent has access to all the information about the concept. However, to actually read a concept from the inbox, we force the receiving agent to reconstruct the concept from scratch using the construction history of the concept. The disadvantage to this is the additional time spent repeating constructions. However, as each agent may be working with different examples from the theory, the representation of a concept sent by an agent may not convey all the information required by the receiving agent. For example, if one agent was working with the numbers 1 to 10, and received a concept from an agent working with 1 to 5, it would effectively have to construct the concept from scratch to fill in the missing details.

As each agent measures interestingness in a different way, an agent receiving a concept will either have to accept the judgement of the sending agent, or re-assess the concept on its own terms. For a heuristic search to perform correctly, the last option is preferable, and so each agent assesses a communicated concept itself and places it in the appropriate place in the agenda. As some of the measures are built up as the concept is constructed, the simplest way for a concept to be re-assessed correctly is to build it from scratch. Therefore, another advantage to reconstructing concepts is that each is properly assessed and incorporated into the theory correctly.

Each agent passes *all* of its concepts to the other agents. At present, whenever an agent sends its concepts, it takes the opportunity to read the concepts in its inboxes. This model may change in future, as we test whether the agency is more efficient if agents send concepts more often than they read them. Whereas all concepts are sent to inboxes, agents are very selective about which concepts they take from the inboxes. The selection procedure is dependent on the task set for the agency, so we discuss this in the context of the problem being addressed in §4.

There are only a few settings available to the user to fine tune the action of the agency:

- The user can choose how many agents to use.
- They can set the search parameters differently for each agent, so that they perform different searches.
- They can detail how each agent selects from its inboxes - including specifying the agents it takes concepts from and which concepts to take.
- They can specify when the sending and reading of concepts takes place. This is specified in terms of how many theory formation steps occur before the agent communicates the concepts it has found (and reads the concepts in its inboxes).

4 Exhaustive Categorisation

When HR is asked to explore a domain it must find out as much information about that domain as possible. In this mode it is difficult to assess how well the program is doing. Certainly it is impossible to find all the concepts in a domain and even in a depth limited search, there are often too many concepts for a program to conceivably find in a reasonable time limit. Also, it is very difficult to assess how creative a program has been in constructing a particular theory. The exhaustive categorisation problem discussed here provides ways to measure the success and creativity of a theory formation program.

4.1 Problem Description and Motivation

We say that a set of examples has been **exhaustively categorised** by a theory if for any possible way to categorise the examples, there is at least one concept in the theory

which achieves that categorisation. For example, given the integers 1 to 4 as examples for number theory, the entire set of categorisations for these integers is:

[1, 2, 3, 4]	[1, 2, 3], [4]	[1, 2, 4], [3]
[1, 2], [3, 4]	[1, 2], [3], [4]	[1, 3, 4], [2]
[1, 3], [2, 4]	[1, 3], [2], [4]	[1, 4], [2, 3]
[1, 4], [2], [3]	[1], [2, 3, 4]	[1], [2, 3], [4]
[1], [2, 4], [3]	[1], [2], [3, 4]	[1], [2], [3], [4]

Once the objects of interest supplied by the user have been exhaustively categorised, a milestone has been passed because the program has learned an answer to any question of the form "Why are x, y and z the same but different to a, b and c". For example, if we asked why 1 and 4 are the same, but different to 2 and 3, any program which had invented the notion of square numbers could answer that 1 and 4 are squares but 2 and 3 are not. At present, we have achieved an exhaustive categorisation of the integers 1 to 5 using HR. This leads us to the problem description: to exhaustively categorise the integers 1 to 6.

4.2 Measuring Success

The number of ways of categorising a set of n objects is defined as the nth Bell number, Bell (1934). The Bell numbers are: $1, 2, 5, 15, 52, 203, 877, 4140, \ldots$ Therefore to exhaustively categorise, say, the integers 1 to 5, HR would need to find concepts which categorised them in 52 different ways. The number of categorisations achieved is some measure of the usefulness of the theory formed and hence of the success of the program.

We introduce the following way to compare two theory formation systems:

Suppose systems A and B both start with the same set of examples and perform the same number of theory formation steps. We say that A is more **creative** than B if the theory it has produced has achieved more categorisations of the examples than the theory produced by B.

There have been entire conferences devoted to understanding creativity in humans and machines,[1] and we are still far from an explanation which could be turned into concrete ways to measure the creativity of a computer program. We believe that a program which produces 100 different categorisations of a set of objects in 500 steps is more creative than one which produces only 10, and this is how we choose to compare the creativity of agencies. We certainly do not claim to have captured the very essence of creativity with these measures. Note that because each agent must reconstruct any concept it decides to read, these reconstruction steps count as theory formation steps. Therefore an agency does not get any steps for free, and the creativity measure is valid.

[1] For example the International Congress on Discovery and Creativity, Ghent 1998.

We hypothesise that employing an agency will improve the creativity of HR. We test this hypothesis in experiment 1. In experiment 2, we assess whether we can improve efficiency without losing creativity. In experiment 3 we assess whether the increase in creativity of an agency compensates for the loss of efficiency due to the communication overheads.

4.3 Experiment 1 - Creativity

We used four agents named:

(H)ardy, (R)amanujan, (L)ittlewood and (W)right.[2]

Each agent worked with the numbers 1 to 6, and employed a different search strategy:

- Hardy - Unary first search
- Ramanujan - Depth first search
- Littlewood - Best first search based on novelty only
- Wright - Breadth first search

The only shared resource was the set of 203 categorisations of the numbers 1 to 6 which are calculated beforehand. Each agent removes a categorisation from the set if it is the first to find a concept which achieves that categorisation. The watcher records which agent found each categorisation first.

We experimented with the criteria by which an agent chooses concepts from its inboxes. We first allowed each agent to read every concept produced by every other agent, but as expected, there was so much repetition of work that the agencies fared much worse than HR running alone. A better alternative is to only allow concepts into the theory which are new to the agent. However, the only way to tell that a concept is new is to test the datatable against all those already in the theory, which is time consuming.

Finally, we realised that for this problem, the most natural selection procedure is to only reconstruct concepts which produce a categorisation which is novel *for the receiving agent.* Because all information about a concept is available, and the agents are working with the same set of examples, it is very quick to check whether a concept's categorisation has already been found by an agent. Further, choosing concepts with novel categorisations guarantees the novelty of the concept itself, and combination with other concepts in the theory is likely to lead to yet more novel categorisations.

To test our hypothesis we compared the creativity of every agency possible using the four agents. These included all agencies with one agent, named H, R, L and W, all agencies with two agents, named HR, HL, HW, RL, RW and LW, all agencies with three agents, named HRL, HRW, HLW and RLW and the agency with four agents, named HRLW. For each of the agencies with two or more agents, we tested two copies: one with no communication at all, and one with **immediate communication** - ie. after every theory formation step, each agent reported

[2] Four highly collaborative number theorists.

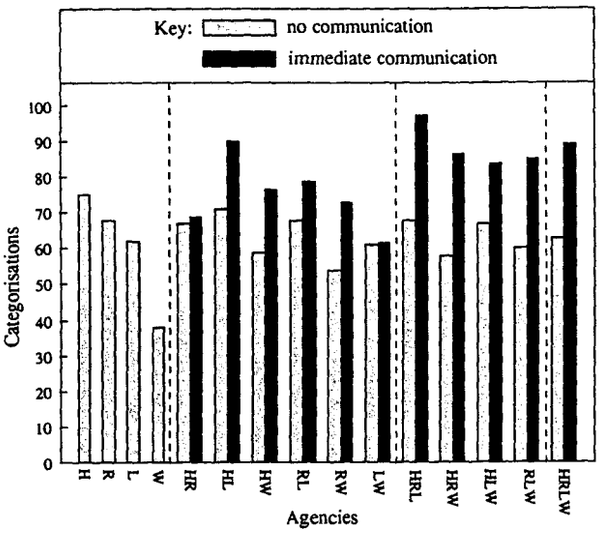


Figure 2: Number of categorisations achieved by agencies after 3000 theory formation steps

new concepts and read those reported by others. We ran each agency for a total of 3000 theory formation steps. In agencies of n agents, each was allowed to perform only $3000/n$ steps. So, in agency HRL, agent H performed 1000 steps as did R and L. We chose 3000 steps because it takes around a minute to perform this number and because 3000 is perfectly divisible by 2, 3 and 4. The search was depth limited to complexity 6, and we ran all tests on a Pentium 500Mhz processor.

Before detailing the results from the test, we report an unexpected phenomenon which occurred when running agencies which communicate. We noticed that the number of categorisations being achieved differed when the program was run with exactly the same settings for the same number of theory formation steps. We are still investigating this, and at present we believe that the Java thread mechanism cannot be trusted to perform exactly the same tasks in the same order. This is a problem because agent L uses a best first search by measuring concepts in relation to the others in the theory. Suppose that agent L read concept C just before it was going to invent a very interesting concept of its own, X. If C was interesting, it would be developed due to the best first nature of the search. Only after C had been developed would X be formed, which leaves less time to develop it. In a different session, if L read concept C just after it invented X, X would be developed before C, giving more time to develop X and the concepts produced from it.

Thus, because our sessions are limited by the number of steps allowed, small changes in the timing of the communication of concepts can make substantial differences in the theories produced. Without explicitly introducing stochastic processes, this models to some extent the way in which luck and serendipity can influence the development of mathematical theories. Imagine the advances which would have been made if Fourier had not lost Galois' manuscript and had saved him from the fatal duel.[3] However interesting the phenomenon is, it makes testing difficult, and we were forced to average the results over 10 sessions to compensate for the difference in theories produced when using a communicating agency.

	expected number of categorisations	
agency size	no communication	immediate communication
1	60.75	n/a
2	63.3	74.75
3	63.25	88.15
4	63	89.2

Table 1: Expected number of categorisations for agencies of different sizes after 3000 steps

Figure 2 shows the number of categorisations found by each agency, with the grey boxes for agencies with no communication, the black boxes for agencies with immediate communication. These results are fairly conclusive. In only two immediate communicating agencies containing a particular agent did the agency containing just that agent perform better (agency HW performed worse than H, and WL performed worse than L).

If we average the scores over agencies of the same size, we can look at the expected number of categorisations for an agency of a given size. As portrayed in table 1, the most creative agency is the immediate communicating agency with 4 agents, which slightly outperforms the average immediate communicating agency with 3 agents. It is also clear that the increase in creativity is due to the communication, not just the fact that the system is using multiple search strategies which cover different areas of the space. In every case the agency with communication outperformed the agency without communication.

4.4 Experiment 2 - Communication Intervals

One way to improve efficiency is to reduce the number of times concepts are communicated and inboxes are checked. With the agencies discussed above, after every theory formation step, if a new concept had been produced it was communicated to the other agents. Similarly, after each step, every agent checked their inboxes for new concepts, and read any which produced a new categorisation. Delaying the reading and communicating of concepts until after, say, every 10th step, will reduce some of the communication overheads. We tested whether delaying the communication would affect the creativity of the agency.

We took the best agency from the first experiment, HRL, and ran it for 100 theory formation sessions. We increased the waiting time for communication from 1 to 10 to 20, etc. up to 1000 steps and recorded the number of categorisations it produced after 3000 steps. Again agents H, R and L each performed 1000 steps. We also recorded the number of categorisations which were **multi-**

[3] See Stewart (1989) for more details of this tragedy.

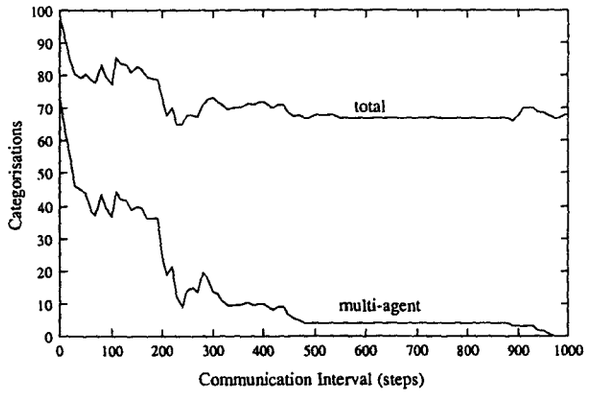


Figure 3: Effect of communication intervals on HRL agency and correlation with multi-agent categorisations

agent. A categorisation is defined to be multi-agent if the first concept which achieved the categorisation had concepts from more than one agent in its construction history. As before, we repeated this experiment 10 times and took an average to counteract the phenomenon described above. Figure 3 shows the effect of lengthening the communication intervals on the total number of categorisations formed and the number of multi-agent categorisations formed.

From figure 3, we see that increasing the communication interval will in general decrease the creativity of the system, and that the total communication scheme outperforms the others significantly. The decrease in creativity is not smooth, however, and we are presently studying the theories produced to explain the peaks and troughs observed in this experiment. The correlation between the total number of categorisations and the number of multi agent categorisations was more pronounced than we expected. Judging by the correspondence in the peaks and troughs on the two graphs, if an opportunity to find multi-agent categorisations is missed, this is not compensated by increased time spent by agents searching on their own.

The decrease in quantity of categorisations is due to our limitation of only 1000 theory formation steps for each agent. For example, agents in the agency where communication occurs after the 600th step only communicate their concepts once, and only those invented before the 600th step. Any interesting concepts it finds after the 600th step are never communicated. This explains the long horizontal sections of the graphs in figure 3 - concepts are communicated too late to be used sufficiently to find multi-agent categorisations. As we see by the end of the total curve, late introduction of the concepts actually hinders the creativity of the agency, and when they are introduced too late to be developed at all, the number of categorisations increases.

The correlation between the number of categorisations and the number of multi-agent categorisations, coupled with the reduction in the number of concepts communicated explains why the creativity of the system declines as the communication interval increases. Therefore we hypothesise that if a system only has a limited number of steps to perform, the smaller the communication interval, the more creative the system will be. More experimentation is required in different theories and with more theory formation steps to confirm this hypothesis.

agency size	expected number of categorisations
1	96
2	114.7
3	110.35
4	116.6

Table 2: Expected number of categorisations for agencies of different sizes after 5 minutes

4.5 Experiment 3 - Efficiency

Having determined which agencies are the most creative, we must assess whether this increases the overall efficiency of the system - the ultimate aim of an agency. Due to the increased overhead in communicating concepts, it may be that agencies can produce more categorisations in a given number of steps, but it takes them longer to carry out those steps because the communication slows down the process. Of course, it is certainly possible to run each agent on a different processor which will give efficiency gains more than compensating for the increased communication overhead. However, it was useful to test the efficiency of the system as a whole.

We scaled the problem up by running each single agent agency and each total communicating agency for a duration of five minutes and comparing the number of categorisations produced. As before, we averaged the results for the communicating agencies over 10 sessions. Finally, we averaged the results over agencies of each size and recorded the results in table 2. We see that the multi-agent agencies are more efficient in general, and we hypothesise that the additional overhead is compensated by the increase in creativity. Again, we plan more experimentation to further investigate this hypothesis.

4.6 Utility and Clarity

To end our investigation of the exhaustive categorisation problem, we looked at the utility of each agent - whether it found any novel categorisation before the other agents. We found that in every run, each agent contributed at least one categorisation to the theory. Figure 4 gives the proportion of categorisations which were first introduced by each agent for a sample 3000 step session using the HRLW agency. It also details the proportion of the categorisations introduced first by each agent which were multi-agent. We see that agent H introduces half of the categorisations. This is due to the effectiveness of its search strategy, and not because H collaborated the most, as we see from the second pie chart that the number of multi-agent categorisations introduced by the agents was roughly equal.

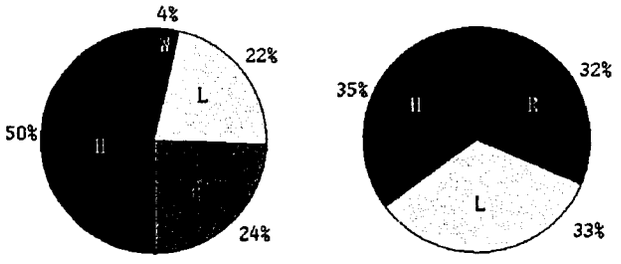


Figure 4: Proportion of (i) all categorisations and (ii) multi-agent categorisations found by agents in an example session with agency HRLW

Agent W does not perform well on this problem and it seems clear that some aspect of a depth first search needs to be incorporated. Also, agent W puts concepts it reads from other agents to the bottom of the agenda, which explains why it produces no multi-agent categorisations. We affectionately call agent W the pedantic agent, as it attempts to cover all possibilities thoroughly while its colleagues race off in many directions. However, agent W performs an important function: it improves the clarity of the theories produced. We can measure the **clarity** of a theory as the average number of theory formation steps required to form a concept from the theory.

A breadth first search will produce the simplest theories, as it will not build concepts of complexity three before building all concepts of complexity two and so on. More than this, as discussed in §2, if a concept is found which matches one already in the theory and the new concept has a more concise definition, the simpler definition is kept. In this way, if agent W reads a concept from another agent and later finds a more concise definition, the theory will benefit from the pedantic approach as clearer definitions for complicated concepts will be produced. We have not compiled the statistics for the gain in clarity of theories obtained when agent W is in the agency, but intend to do so when we run further tests on these agencies.

5 Further Work

Improvements in efficiency could be made by sharing parts of the agenda between agents, because presently an agent sending a concept and an agent receiving the concept will develop it in some identical ways. We also hope to use agencies to improve the modelling of cross domain theory formation as discussed in Steel et al. (2000). At present, to produce cross domain concepts, HR must encourage the combination of two concepts from different domains. Considerations also have to be made in the assessment of the cross domain concepts. We anticipate that using an agent to produce a theory in each different domain, with concepts being communicated between agents and hence across domains will greatly improve the model for cross domain theory formation.

5.1 Extrapolating Integer Sequences

As described in Colton et al. (2000a), HR has been adapted to perform theory formation in a goal based way in order to perform machine learning tasks. It is used to find a concept with examples which match the examples given by the user. This can be applied to the problem of extrapolating integer sequences, where the concept to be learned is a sequence. HR performs a forward chaining search and is equipped with a lookahead mechanism enabling it to spot when it has found the concepts necessary for the definition. For example, given the sequence of odd primes, as soon as HR invents odd numbers and then prime numbers, it looks ahead and notices that their combination will produce the desired concept.

We are currently experimenting with an agency approach to sequence extrapolation, motivated by limitations when extrapolating certain sequences. These sequences highlight some of the difficulties HR faces:

What is the next in these sequences?
- 2,3,5,7,11
- 1,3,6,11,18,29
- 101,102,104,106,110

The first is identified by HR as the sequence of prime numbers. The other sequences cause more difficulty.

If we calculate the difference between successive terms of the second sequence, we get the prime numbers again: 2, 3, 5, 7, 11. In IQ tests where sequence extrapolation problems are common, knowledge of this difference transformation is expected. This transformation is so common that it suggests tailoring HR's forward looking mechanism to look for either the original sequence or the difference sequence. This caused many technical problems, and was a messy solution. Moreover, in future versions of HR we hope to make the search more goal directed, using the examples supplied to explicitly direct the search. In this case, as the original and difference sequence are often so different there will be a conflict in the search strategies. Therefore, we are experimenting with a multi-agent approach to sequence learning, where two agents are employed, one to look for the original sequence and the other to look for the difference sequence. The model is certainly much neater and works well. We are still testing whether multi-agent model is more efficient than the single-agent model, but initial findings are encouraging.

If we now look at the third sequence and take 99 from each term, we get: 2, 3, 5, 7, 11, which is the prime sequence for the third time. This is not, however, a common transformation, and there are too many similar transformation to reasonably dedicate an agent to the output from each one. A better model is to take each concept produced and determine which, if any, transformation from a family of transformations would produce the desired sequence. For example, when HR invents the concept of prime numbers, it could look at the family of transformations which add on a particular number to every term in

the sequence. To get from the first term of the prime sequence it has just invented to the first term of the target sequence, it needs to add on 99. It would then determine that adding 99 is the desired transformation.

We are presently testing whether it is more efficient to have a single agent attempting to identify the correct transformation. This agent takes the concepts from the other agents which could possibly be transformed and attempts to find the correct transformation. Again, the model works well, and we are assessing whether this is efficient. We cannot possibly hope to cover all possible sequences which an inventive person could produce, but we do hope to show that the agent based approach improves the coverage of the system.

6 Conclusions

Adapting HR to employ a multi-agent approach is a natural way to extend its theory formation abilities. Agencies equipped with a method for communication of concepts and selection of concepts better model the way in which collaborative research progresses in science than single programs running in isolation. In Furse (1990), the author proposes a network of theory formation programs each communicating their most interesting concepts to the programs on the network. This is a good model, and similar to the one we have implemented. However, in our model, each agent communicates all its concepts to the others, but will assess a concept on its own terms rather than accepting the assessment of the sending agent. In this way, a concept which seems dull to one agent may be picked up and fruitfully utilised by another.

Machine learning programs such as Progol, Muggleton (1995), are asked to find a single concept which classifies the examples supplied correctly. Therefore, among other ways, success can be measured by the number of concepts it can learn from a predefined set. It is more difficult to perform a quantitative assessment of a theory formation program, because the goal is to find many interesting concepts and obtain some understanding of the domain. We have chosen to measure the number of different ways a set of examples can be categorised by a theory to determine the quality of the theory and accordingly developed a way to compare the creativity of different systems.

We have demonstrated that a multi-agent approach can increase the creativity and efficiency of a system, even before any advantage is gained from running each agent in parallel. Autonomous intelligent agents have emerged over the last decade as an important technique to solve many interesting problems, and improve efficiency in many areas, Jennings and Wooldridge (1997). We have shown that theory formation programs can benefit from an agent based approach. We also hope to demonstrate that the theory formation agencies we plan to implement in the future will apply fruitfully to other areas of artificial intelligence such as machine learning and theorem proving.

Acknowledgements

We wish to thank Edmund Furse for in depth and informed discussions about possibilities for multi-agent theory formation. We would also like to thank the anonymous reviewers for their enthusiastic comments about the extended abstract for this paper, and the symposium chair, Geraint Wiggins, for his effort in bringing us all together. This work is supported by EPSRC grant GR/M98012.

References

E Bell. Exponential numbers. *American Mathematics Monthly*, 41:411–419, 1934.

S Colton, A Bundy, and T Walsh. HR: Automatic concept formation in pure mathematics. In *Proceedings of the 16th IJCAI*, 1999.

S Colton, A Bundy, and T Walsh. Automatic identification of mathematical concepts. In *Machine Learning: Proc. of the 17th International. Conference*, 2000a.

S Colton, A Bundy, and T Walsh. On the notion of interestingness in automated mathematical discovery. *IJHCS*, Forthcoming, 2000b.

R Davis and D Lenat. *Knowledge-Based Systems in Artificial Intelligence*. McGraw-Hill Advanced Computer Science Series, 1982.

S Epstein. Learning and discovery: One system's search for mathematical knowledge. *Computational Intelligence*, 4(1):42–53, 1988.

E Furse. Why did AM run out of steam? Technical Report CS-90-4, Department of Computer Studies, University of Glamorgan, 1990.

N Jennings and M Wooldridge. *Agent Technology: Foundations, Applications and Markets*. Springer, 1997.

T Kuhn. *The Structure of Scientific Revolutions*. University of Chicago Press, 1970.

D Lenat. Eurisko: A program which learns new heuristics and domain concepts. *Artificial Intelligence*, 21, 1983.

S Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13:245–286, 1995.

M Sims and J Bresina. Discovering mathematical operator definitions. In *Machine Learning: Proc. of the 6th International Conference*. Morgan Kaufmann, 1989.

G Steel, S Colton, A Bundy, and T Walsh. Cross domain mathematical concept formation. In *Proceedings of the AISB-00 Symposium on Creative & Cultural Aspects and Applications of AI & Cognitive Science*, 2000.

I Stewart. *Galois Theory*. Chapman and Hall Mathematics, 1989.

Knowledge-based Composition of Classical Minuets by a Computer

Mathis Löthe
Universität Stuttgart; Institut für Informatik Breitwiesenstr. 20-22 D-70565 Stuttgart, Germany
Mathis.Loethe@informatik.uni-stuttgart.de

Abstract

Composition of piano minuets in early classical style takes an intermediate position between restricted and free problems of computerized composition. One of its distinct features is that microstructure (i.e. relations between notes), macrostructure (i.e. relations between larger parts), and their interdependencies have to be dealt with. In this paper, we describe a method for rule-based note by note composition of minuet melodies, which considers harmonic and structural descriptions given as prerequisite. We also present 16-bar examples composed by the algorithm in the chord-based and the scale-based melody type with different parameter settings. The algorithm uses explicit knowledge about the composition. The paper describes sources of knowledge such as literature (especially period (historic) literature) and human experts.

1 Overview

This paper describes a knowledge based approach to composition of minuets[1] for piano in early classical style (that is the period of the young W.A. Mozart) by a computer. The goal is, to approximate the style of this period as closely as possible. The system should only generate *stylistically correct* minuets, which are pieces an expert would accept as a classical minuet. On the other side, it is allowed to undergenerate, that means there are perfectly acceptable original minuets, which are not covered by the systems knowledge.

The key problem is to get explicit musical knowledge, which is both formally represented and computationally useful. With this knowledge, the composition system can be built by using different AI programming techniques.

This paper describes work in progress. While some parts of the composition system are already realized, but have to be improved, others are still missing completely.

The result of the composition process is music represented in pitches and durations. It is the same information, classical composers wrote down in staff notation, neglecting the fact, that they sometimes added dynamic and articulatory signs. The interpretation of the music, the sound synthesis, is also not dealt with here.

The composition of minuets can be discussed at two levels. On the level of *microstructure*, relations between notes in horizontal direction (melodic intervals) and in vertical (synchronous) direction (harmony) are discussed. *Macrostructure* deals with the larger structural parts in horizontal direction, e.g. the length of phrases and their thematic and tonal relations.

[1] a genre of dance music from the 17th and 18th century in triple meter

The vertical (synchronous) direction is described by harmony. Voice leading rules (e.g. the *forbidden parallel fifth*)-rule combine the vertical and microstructurally horizontal direction.

The process of classical minuet composition can be divided into several subtasks: Using a top-down-approach, it starts with planning the macrostructure of the piece. The next step is planning the harmonic progression inside a phrase, defining the basic parameters of the melody, and composing it. Finally the minuet can be completed by adding the bass (and other accompaniment).

Here, these subtasks are described in a linear order. Though it might be possible to compose by strictly following this order, human composers usually mix the subtasks. They revise decisions of earlier subtasks while working on a later subtaks to avoid problems. Experienced composers are able to work on one subtask and to consider the basic needs of the later subtasks (without really doing them, i.e. figuring out all details). In spite of this, literature describes the different subtasks separately.

2 Knowledge Sources

The author of a knowledge-based AI system has to acquire the domain knowledge, has to formalize it and has to provide an algorithm which the computer can use to find a solution. Often, it is helpful to look at the ways human experts codify their knowledge, e.g. their literature. Explicit knowledge about composition is needed for teaching composition students or for scientific works – e.g. for comparisons between styles or for classification of borderline cases of works. Human composers learn the subtasks of composition part by imitation: *"It should sound*

like this", part by explicit rules (and doing exercises). The mixture is different for different subtasks. Melody composition is learned more by imitation, voice leading more by explicit rules.

For the field of classical minuets, the following knowledge sources are considered:

1. The usage of period literature[2] like Riepel (1752) and Koch (1782) or modern literature with an historic approach Budday (1983) ensures a correct view of the style.
2. Ahistoric theories such as theory of harmony describe properties universal to all tonal music. Ahistoric theories are usually more formalized and therefore easier to implement than knowledge from period literature.
3. Asking experts (musicologists, composition teachers, and composers) in the style can yield explicit knowledge which might not have been written down.
4. Manual analysis can either give an abstract verbal description of a minuet or it can search minuets for evidence in favour or against a specific hypothetical rule.
5. Computer analysis of works can help with the second form of analysis described above: the search for evidence for or against a rule.
6. Computer composition experiments: The composition algorithm is run with a ruleset including a hypothetic rule and then with another ruleset without that particular rule. The results are compared (or judged by an expert).
7. Cognitive introspection. Ask someone, who has just composed a minuet, how and why he made certain decisions.

The last four knowledge sources should give access to implicit knowledge, which is not available in literature.

3 Principles

A typical feature of classical compositions is, that there are a lot of relations between elements of the music. The representation has to provide the possibility to express them and has to make those relations accessible for the composition process.

3.1 The Score Chart

The composition strategy is based on a representation of a musical score in a *score chart*, which represents the information classical composers wrote down. Notes are stored with pitch and duration. On top of this, the score chart also stores abstract information for and about the composition process in form of *structural elements*.

One example for abstract information is the harmonic base, which is stored by degree (as in degree theory) relative to the current key. The current key (which changes during the piece, when the piece is modulating) is stored relative to a base key, which is fixed for the whole chart.

Structural regions contain abstract descriptions about a time interval (region, section) of the piece. Examples are a cadence, a transition in a particular voice, or the similarity between a region and another region. The chart provides primitives to access this information. This allows the composition function to refer to all notes, which are already composed and to consider the meta information.

3.2 Macrostructure Descriptions

The macrostructure of minuets is thouroughly discussed in period literature. The basic unit of a composition is the *sentence*[3] The *bar order* and *tonal order* [4] describes the relations of the lengths of the sentences and the end formulas of the sentences in a piece. The end formula of a sentence can be either a full or a weak cadence and both can be in the root or the fifth degree[5].

Different combinations are possible. A prototypical minuet Budday (1983) consists of four sentences with a length of 4 bars each.

Bars 1-4 end with a weak cadence on the tonic.

Bars 5-8 modulate to the key of the dominant and close the first half of the piece with a full cadence on the fifth (relative to base key).

Bars 9-12 have to lead back to the base key and frequently have more distant harmony.

Riepel (1752) describes two popular harmonic and melodic patterns for this sentence. The base of both is a two bar motive, whose harmonic base is an intermediate dominant[6] with its resolution. The sentence consists of the motive and its *real sequence*. A sequence of a motive means moving it to a higher or lower pitch while keeping the intervals. In a *real* sequence, the exact interval size is adjusted to harmony, e.g. a major third can become a minor third if it is necessary to fit into harmony (as opposed to a chromatic sequence, where interval sizes stay exactly the same).

[2] In musicology, *period* or *contemporary* literature (zeitgenössische Literatur) is literature written in the same time period as the musical period or style it describes.

[3] Original: *Satz*. Another english term is *phrase*. *Period* is sometimes used for one sentence, sometimes for a group of sentences.

[4] Original: *Taktordnung* and *Tonordnung*

[5] Original: *Grund-* und *Quintkadenz*, *Grund-* und *Quintabsatz*

[6] It is a dominant (fifth degree) relative to the chord, which is following directly. In classical style, the dominant seventh chord with its higher tension was popular as intermediate dominant.

In the *Monte*[7]-sequence, the first occurence of the motive ends on the fourth degree. It is then sequenced a second up and the sentence thus closes with a weak cadence on the fifth degree. In G-major, the harmonic base for a Monte is $G^7 - C - A^7 - D$.

In the *Fonte*[8]-sequence, the motive is sequenced a second down. The first occurrence of the motive ends on the second degree and is sequenced down to close the sentence with a weak cadence on the root. In G-major, the harmonic base for a Fonte is $E^7 - a - D^7 - G$

Bars 13-16 close the minuet with a strong cadence on the root.

In the current state of the composition system, the macrostructure description consists of a set of parameters for the piece and a sequence of sentence descriptions. A sentence description contains elements such as: the end formula of the sentence, the harmonic base for the sentence, and those parameters, which differ from the default parameters of the piece. Another important element of a macrostructural description is a correspondence (similarity, relation) between a section of the sentence – the *correspondent* section – and a preceding section of the piece (in the same or a previous sentence), the *model*. The monte- and fonte-sequences are represented as a correspondence.

A systematic collection of possible end formulas in different meters can be found in Budday (1983). There, the end formula patterns are written down in notes as melody and bass. The harmony is indicated by figuration of the bass. The end formulas in Budday (1983) are not expected to occur verbatim in pieces, but each pattern is an abstract description for a class of end formulas. The patterns are written down in C-major, but they can occur transposed to any key. Thus they are meant as degrees in respect to a key, not as absolute pitches. A pattern for 2/4-meter can be applied to a piece in 2/8 meter and a two bar pattern in 2/4 meter can match fit to one bar in a piece in 4/4 meter. That means, the pattern represents relative levels of metric stress rather than absolute durations. Middle voices might be present in a piece. The figuration of the bass in the pattern specifies the harmony, into which the middle voices have to fit.

The information described by an end formula pattern in Budday (1983) is the sequence of harmonies, their respective metric weight, and the basic leading of outer voices (still allowing embellishments or other notes without structural importance). The end formula patterns in the composition system also represent this information.

3.3 Rule-based Melody Composition

A macrostructural sentence of the melody is composed note by note. Base of the note-by-note composition is the *alternative function*, which computes the set of alternatives to continue the melody by one note. To allow more flexibility at experimenting, this function has been split up into a sequence of rules where each rule deals with a certain musical phenomenon.

A rule is applicable, if its precondition is satisfied. If the rule is applicable, the consequence of the rule adds, removes, or modifies alternatives from a set of alternatives according. The precondition and the consequence of a rule have access to the score chart containing the current state of the composition and the current set of alternatives. The alternative function applies one rule after the other in the given order.

To facilitate programming the precondition and consequence of a rule, an alternative is represented as a combination of interval, duration of the note to be added, and a list of new structural elements. The latter is necessary to express the fact, that this alternative is only correct, if the notes following later satisfy certain conditions. The structural element in the chart ensures that the composition process considers those conditions when it composes the following notes.

There are two basic types of classical melodies:

Scale-based melodies consist mainly of seconds. Leading notes must be observed: a note of degree 7 must be led a second up, a note of degree 4 on a dominant harmony (the seventh of the chord) must be led a second down. Experiments with differerent larger intevals werde made. The rulesets used for the examples in this paper allow thirds, if they do not follow each other immediately.

In addition to ordinary melody notes, which are members of the current harmonic chord, some non-harmonic tones are possible. They have to be prepared and resolved in a specific way and must therefore be represented as structural elements to ensure proper treatment. Currently only passing tones are implemented.

Chord-based melodies were quite popular in classical style because they put more emphasis on harmony. They consist of steps to the next note of the harmonic chord. Chord sections can be joined in different ways. Currently only a join by a second is implemented.

In both melody types, repetitions (two notes with the same pitch) are allowed. Because repetions have some kind of emphasizing effect, two repetitions must not follow immediately and there are rhythmical restrictions in order to avoid a weird distribution of stress.

As described in the last section, structural elements can be determined top-down by the macrostructure description, e.g the type of end formula for a sentence. But they can also be *"found"* bottom up during melody composition and inserted via the structural element slot of the alternative like the passing notes. Another case, where

[7] *Monte*: ital./lat. up the hill

[8] *Fonte*: ital./lat. down into the well

a structural element overrides the default rules is the *3-quarter-bar*. It allows a bar, which consists of three quarters of the same pitch in spite of the default repetition limits. A 3-quarter-bar is typical for the genre of minuets and can be frequently observed in original compositions.

The correspondence mentioned above is required top-down by the macrostructure description. The correspondence is represented as a structural element for the correspondent section and has the location of its model section and the variation operator as arguments. Similarity and variation can be frequently observed in classical melodies. The varied correspondent section keeps most properties of its model (in case of a sequence: intervals and durations) and modifies others (in case of the sequence: start note and therefore the absolute pitch).

3.4 Composition Strategy

The melody composition rules constitute a search problem as described by Russell and Norvig (1995), chapter 3. The state is the current state of the composition. The successor function, which determines the set of possible successor states, is the combination of the rules. A goal predicate for successful melodies and a fail predicate are defined. When the problem is defined in those terms, it is possible to switch between different search strategies without reprogramming the operators.

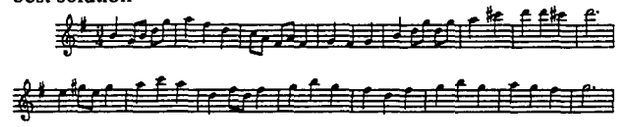
There have been experiments with different search strategies, including a chronological backtracking strategy with random selection of the successor state and a random shot strategy. With the current ruleset, a sentence-wise composition using a random-shot strategy for the sentences, was most suitable. With the random shot strategy the successor state is selected by random. In case of a fail or no successor states, the next attempt starts with the start state (the beginning of the sentence) again. The random shot strategy works well, if the percentage of solutions in the search space is not too small. The backtracking strategy becomes inefficient, if an early wrong decision can be recognized as wrong (leading to a fail) only much later. In this case, the backtracking strategy does a complete search on branches, which do not contain a solution.

Following the macrostructural description, the sentences are composed from front to end. For the first sentence, the random shot strategy composes the desired number of solutions. For the next sentence, the given number of solutions is composed for each solution of the previous sentence. The composition functions have access to the macrostructure description and to the sentences already composed.

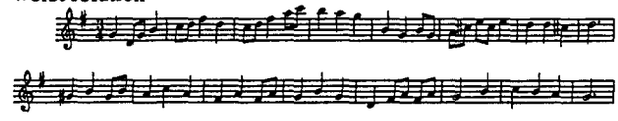
The composition system has been implemented in Common Lisp, using CLOS classes for score chart elements such as notes, structural elements, harmonic elements, and other musical entities. Other concepts of the program e.g. the search states of the melody composition, the alternative, and the rules are also represented as CLOS classes.

4 Experiments

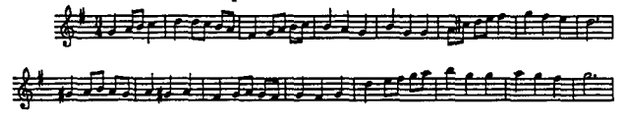
1a. chord-b. *Fonte*-seq. to weak cad. on root in bar 12.
best solution



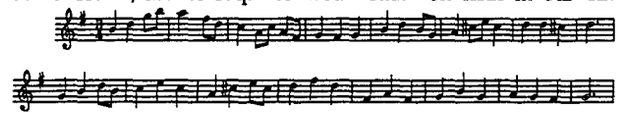
1b. chord-b., *Fonte*-seq. to weak cad. on root in bar 12.
worst solution



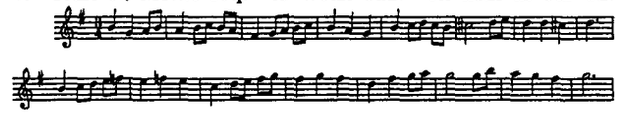
2. scale-b., *Fonte*-seq. to weak cad. on root in bar 12.



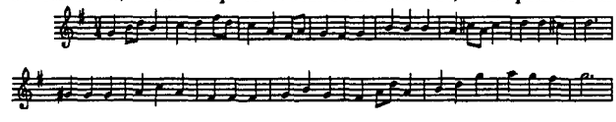
3. chord-b., *Monte*-seq. to weak cad. on fifth in bar 12.



4. scale-b., *Monte*-seq. to weak cad. on fifth in bar 12.



5. chord-b., *Fonte*-seq. to weak cad. on root, + 3-quarter-bar



6. scale-b., *Fonte*-seq. to weak cad. on root, + 3-quarter-bar

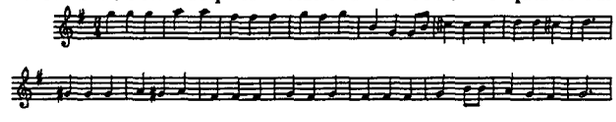


Figure 1: Computer composed minuets

Figure 1 shows examples of computer composed minuets. They all use the prototypical 16-bar, 4 sentence macrostructure described in section 3.2 with different parameters for the melody composition. For each sentence, two alternatives were composed. Therefore, each experiment yields 16 minuets, from which one was selected. Example 1a and Example 1b are results of the same experiment. Nr. 1a was judged by an expert[9] as best of the

[9] Thanks to Georg Wöetzer from the *Staatliche Hochschule für Musik*

16 minuets in the result set, Nr. 1b was judged as worst. Both melodies contain passages, which are perceived as clumsy or aimless without being really wrong. In melody 1a the repetition in bar 5 sounds aimless. The second line of 1a is simple, but quite good, because it outlines harmony well and many of the bars follow a common pattern. They have the same note on beat 1 and on beat 3, while jumping a third up or down on beat 2. Having common structural roots is good for a classical melody.

Melody 1b (judged as worst) contains just more of the clumsy passages than 1a. The difference between good and bad solutions in a solution set composed by this system is rather small. For the best known approach to style-imitating composition by a computer – the system of David Cope – the best results are really good, but a much larger difference in quality between the best and worst solutions is reported [Cope (1995)].

The following examples show different combinations of the fonte- and the monte-sequence with the scale based and the chord based type of melody. For some sentences of the chord based melody type, scale based end formulas were allowed. The last two examples make use of the 3-quarter-bar structural element (Nr. 1 to 4 do not). In the current state of the system, the chord-based melodies tend to be better than the scale based, because chord based melodies with their well perceivable harmonic tensions are more robust against the aimlessness and clumsiness caused by the random decisions.

A statistic of the search process has been made for experiment 1a/1b, which has a chord based melody and for experiment 2, which is scale based. The statistic should give more information about the structure of the search space.

Example number (as in Fig 1) Based on		1a/1b chord	2 scale
Solutions (comp.sent.)	total	30	30
Random shots	total	9294	3956
Shots per solution	avg.	309.80	131.86
Length of solution	avg.	12.43	13.10
Length of shots	avg.	6.77	5.64
Search states	total	67477	25843
Branching factor	avg.	3.19	3.09

The number of *solutions* is the total number of sentences composed for the 4 sentences of the macrostructure. They are the successful attemps, while the number of *random shots* contains both successful and unsuccessful attempts. An unsuccessful attempt has either died out – i.e. there were no successor states – or it has failed (early or at the end of the specified length of 4 bars) because the fail-predicate of the search strategy detected an error. The average number of *shots per solution* (i.e. attempts per success) shows how difficult composition with a certain parameter set and a certain macrostructural description is. Separate statistics for the 4 sentences of the macrostructure show, that the third sentence, which contains a Fonte or Monte sequence, needs a higher number of attempts per solution than the other 3 sentences.

The *length of solution* is the average number of notes in a solution while the *length of shots* is the average number of notes for all attempts (successful and unsuccessful). The total number of *search states* is an indicator for the asymptotic complexity caused by the ruleset. Looking at one call to the successor function, the difference between two rulesets is just a constant factor. An difference in asymptotic complexity can be caused by the influence of the ruleset on the search space. This difference can be observed through the number of *search states*. The *branching factor* is the average number of successors for each state.

und Darstellende Kunst, Stuttgart (Stuttgart State Conservatory)

5 Summary

Being work in progress, there are many possibilities for improvement. The composed melodies can be made more interesting by defining more types of non-harmonic tones and by analyzing and selecting melodies for qualities beyond harmonic and melodic composition rules e.g. musical tension.

The other subtasks have still to be realized. They include the generation of macrostructural descriptions and harmonic progression schemes, which both are currently written by hand, and the composition of the bass, which is still missing completely.

In the past, one class of approaches to automated tonal composition have focussed on composition problems on a microstructural level with firm restrictions such as chorale harmonization Ebcioglu (1986) or 16th century counterpoint Schottstaedt (1989). On the other side, there are systems for algorithmic composition such as Taube (1994), which can be used to compose on different structural levels, but which do not deal with tonal restrictions. The sonata composition system Berggren (1995) considers focusses on macrostructure because it starts with predefined building blocks.

The distinct features of minuet composition are its middle position between free and restricted tasks, the integration of different structural levels and the fact that it produces tonal music. The early classical style and the genre of minuets is defined by an extensive cultural context. The system therefore needs a lot of musical knowledge. It is also a goal of the research, to document the sources for each element of the system's knowledge.

References

Ulf Berggren. *Ars combinatoria – Algorithmic Construction of Sonata Movements by Means of Building Blocks Derived from W.A.Mozart's Piano Sonatas*. Dissertation, Uppsala University, 1995. 174 S.

Wolfgang Budday. *Grundlagen musikalischer Formen der Wiener Klassik: an Hand der zeitgenössischen Theorie von Joseph Riepel u. Heinrich Christoph Koch dargest. an Menuetten u. Sonatensätzen (1750 - 1790).* Bärenreiter Verlag, Kassel, 1983.

David Cope. *Experiments in musical intelligence.* The computer music and digital audio series ; 12. A-R Editions, Madison, Wisc., 1995. Enthält: 1 CD-ROM, XIV, 263 S.

Kemal Ebcioglu. An expert system for harmonization of chorales in the style of J.S.Bach. Technical Report 86-09, Department of Computer Science, University of Buffalo - State University of New York, 1986.

Heinrich Christoph Koch. *Versuch einer Anleitung zur Composition, 3 Bände entstanden 1782-1793.* Reprint 1969 Georg Olms Verlag Hildesheim, 1782.

Joseph Riepel. *Sämtliche Schriften zur Musiktheorie (entstanden 1752-1782), Ed. Thomas Emmerich.* Reprint 1996 Böhlau Verlag, Wien, Köln, Weimar, 1752.

Stuart J. Russell and Peter Norvig. *Artificial intelligence : a modern approach.* Prentice Hall series in artificial intelligence. Prentice Hall, Englewood Cliffs, NJ, 1995. XXVIII, 932 S.

William Schottstaedt. Automatic counterpoint. In Max V. Mathews and John R. Pierce, editors, *Current Directions in Computer Music Research,* System Development Foundation Benchmark. MIT Press, 1989.

Heinrich Taube. Common music. Technical report, Zentrum für Kunst und Medientechnologie, Karlsruhe, Germany, 1994. available at `ccrma-ftp.stanford.edu:/pub/Lisp/cm.tar.Z`.

II. Modelling & Supporting Creative Processes

Reciprocal models in cooperative problem-solving

John Cook and Mark Springett
School of Informatics and Multimedia Technology, University of North London
2-16 Eden Grove, London N7 8EA
j.cook@unl.ac.uk; m.springett@unl.ac.uk

Abstract

The aim of the work described in this paper is to provide a basis for addressing two pedagogical agent design issues: (i) the need to detect and diagnose the users current needs, and (ii) the need to establish the most appropriate form that a pedagogical agent intervention should take. We describe a taxonomy of effects, referred to as 'reciprocal models' that address both of these issues. This is illustrated using examples of creative, cooperative problem-solving in musical composition using the MetaMuse system. The implications for the design of intelligent computer-based agents is then discussed.

1 Introduction

This paper is a contribution to research into design of pedagogical agents in educational software. The intention is to extract lessons from the study of human-human and human-computer dialogue in order to inform design. Design of agent-based systems faces two distinct issues. One is the need to detect and diagnose the users current needs, by reasoning about their behaviour. The other is the need to establish the most appropriate form that an educational intervention should take. The latter issue is critical given the risks involved when the system intervenes in creative or constructive task performance. This paper describes work towards providing an analysis framework that can address these difficult design issues.

The study of cooperative problem-solving dialogues is used here to analyse the type of effect that the mentor or agent's contribution may have on the user(s). It describes a taxonomy of effects, referred to as 'reciprocal models' (RM). These describe reactions to dialogue turns observed during empirical studies. The aim is to provide a basis for addressing the two issues described above. This analysis contributes to the detection of user needs through linking intervention to types of user behaviour. The seven types of RM identified as results in this paper add to our understanding of the learning processes involved in a creative, cooperative problem-solving situation. Furthermore, our study also serves as a preliminary analysis that can inform design of actual representations in the pedagogical agent.

Whilst there is a sizeable body of literature on learning and cognition, less is known about the cognitive effects of cooperative learning in a creative context. Unlike more formalised domains, e.g. physics and mathematics, the nature of dialogues that result from cooperative learning in open domains remains unclear. However, when we learn the ability to solve a creative problem, dialogue with a teacher or with other learners can play an important role as part of an interactive learning mechanism (Lipman, 1991).

2 Reciprocal Models

The authors of this paper were particularly interested in developing models that can be used as a basis for the design of pedagogical agents. In the paper we briefly describe a pedagogical agent called MetaMuse (Cook, 1999) that was designed to 'structure interactions' in such a way that would, it was predicted, facilitate creative problem-solving dialogues. The main result presented in this paper is a detailed elaboration of "Reciprocal Models" (RM), which is a taxonomy of effects. RM can have different types. These types can have an influence (both positive and negative) on the nature of a situation in which two individuals are involved in cooperative problem-solving. In one example of reciprocal models, given in Cook (1998), post-experimental interviews revealed in one case that once the teacher's expectations had been inferred by the learner (i.e. beliefs about a partner's beliefs), the approach taken by that learner seemed to be one of trying confound those expectations, to surprise. This paper concludes with a brief discussion of how our model of RM can be generalised to a re-implementation of MetaMuse.

3 Example of a cooperative problem-solving dialogue

In order to illustrate more fully what we mean by cooperative problem-solving, Table 1 gives an example. Table 1 shows dialogue which is taken from Session 4 of the study described below. This pair (who are equal partners engaged in solving a creative problem) have just finish a whole cycle of explaining their creative idea (two musical phrases). The pair are discussing whether to leave their current idea behind. Subject 7 makes a suggestion that they could build on what they already have (i.e. refine it) and makes some concrete suggestions of either 'tying their two phrases together' or even 'playing the high phrase in a different manner'. Subject 8 picks up on the second idea (a creative intention) of changing the 'rising part' (i.e. the high phrase). Subject 7 ends the extract shown in Table 1 by confirming that their creative intention "would make sense".

Table 1: Example of cooperative problem-solving

Utterance
Subject 8: ... and [TYPES: "REPEAT THAT MOTIF TO"] to metamorphose into something slightly different. **Subject 7:** And good. Yeah hard to get some sort of **Subject 8:** [CONTINUES TO TYPE: "METAMORPHOSE INTO SOMETHING"] I think that's what we're saying [MORE TYPING BY SUBJECT 8: "SLIGHTLY DIFFERENT.", CLICKS RETURN, SOME COMMENT TO EXPERIMENTER.] Shall we just leave it. I'll just leave that yeah [CLICKS ON 'CANCEL' AND THEN 'MOVE ON'] analyse-diagnose the phrase that's **Subject 7:** Where does it actually ask you to do that, you could either try to tie the two bits together. With that you could actually try to make the high phrase played in a slightly different manner. **Subject 8:** Yeah. I think you, you're kinda making the most of the rising part in the, in the bottom in the **Subject 7:** Hum. **Subject 8:** Lower end and **Subject 7:** Yeah that would make sense.

4 Metamuse and ealier study

MetaMuse (Cook, 1999) is a pedagogical agent that has been designed to structure interactions between pairs of cooperating learners. MetaMuse can only deal with a small task revolving around the chromatic transposition of a four note phrase. The pedagogical agent is capable of certain types of symbolic reasoning based on its knowledge. For example, the pedagogical agent 'knows' that 'an interval leap of 23 or more may indicate a musical phrase boundary'. MetaMuse has been subjected to preliminary evaluation with six pairs of cooperating learners (Cook and Smith, 1999). This initial evaluation, which involved the use of a questionnaire and post-experimental interviews[1], obtained a favourable results. However, the initial evaluation of the teaching agent did not give us any insight into the interactive means by which learning agents engage in cooperative problem-solving. The remainder of this paper describes empirically based dialogue analysis and modelling that explored the above question.

5 The Study

Six sessions were conducted that involved pairs of cooperating learners interacting with each other and MetaMuse. The twelve participants ranged from undergraduate students, postgraduate students, research fellows and members of staff (teaching and support). Seven students were male and five female. Each learner-learner-MetaMuse session lasted between 30 and 40 minutes and was recorded on two video cameras. One camera focused on the computer monitor and the second camera was pointed at the study participants (dyads).

Each session involved the participants being asked to work together in order to carry out a small composition task. Briefly, the compositional task was for the participants to attempt, using MetaMuse, to create a phrase by the repeated chromatic transposition of an initial four note motive (C C# F# G). Participants were not given any instructions on how to cooperate on the task other than being requested to "work together on the task". Following each session the observers of the session compared notes and decided on which post-experimental cues to use. Each participant was then individually interviewed for 10 to 15 minutes. Approximately three hours

[1] Ericsson and Simon (1993, p. xlix) have suggested that following the completion of a task, cueing distinct 'thought episodes' is a useful way to approach the gathering retrospective verbal reports. This involves constraining the retrospective report by the subject to the recall of distinct thought episodes.

of learner-learner interactions were gathered. Three hours of post-experimental interview data was also collected and extracts incorporated into the analysis. The dialogue analysis described below was performed with the assistance of computer-assisted qualitative methods (NVivo). If a dialogue interaction appeared to involve RM, then that segment of interaction was coded using NVivo. This is an evolutionary approach to dialogue analysis, i.e. the results described below evolved as the analysis progressed.

6 Result and interpretations

The main result is reported below with a brief discussion given in the context. The main empirical result was the seven types of RM. The seven types are characteristics of that educational process, and can have an influence on the nature of a situation in which two individuals are involved in cooperative problem-solving.

The total column in Table 2 shows that on 69 occasions (in the six sessions AND the interview data) evidence was found to support the existence of RM.

Table 2: Reciprocal Model dimension scores by session

	Session						
Category	S1	S2	S3	S4	S5	S6	Total
Affect	0	1	3	0	0	0	4
Attention	0	1	0	0	1	1	3
Creative intention	0	1	2	6	2	6	17
Experiment	4	1	1	0	0	1	7
Knowledge level	4	1	5	0	1	0	11
Social status	0	2	3	0	0	0	5
System	11	1	5	0	4	1	22
Total	19	8	19	6	8	9	69

Indeed, as we undertook the process of dialogue analysis our definition of RM (given in the Introduction) was extended to include the building of ones own internal model and the building of models of external agents. Each model (internal and external) can potentially possess seven types. The external model will consequently have an impact on ones own internal model and on ones actions. Following an examination of the dialogue data we found evidence of **seven** types of RM (each type potentially related to other types). The seven types of RM that were identified in the corpus were: affect, attention, creative intention, experiment, knowledge level, social status and system. Below we briefly describe each type of RM and provide selected examples of the types of RM from the corpus.

6.1 Affect

Affect is the disposition one brings to learning (Ashman and Conway, 1997) and includes motivation and emotion. The dialogue extract shown in Table 3 was taken from the interview with subject 5 following the third session.

Table 3: Example of RM affect

Utterance
Subject 5: So sometimes I feel that it's nice to try things and not feel watched by the system it's still a test ..... in terms of the system that it's forcing you to make these decisions and you cancel but you think that that would be the wrong thing to do. **Interviewer:** Did you feel a slight tentativeness a slight nervousness. **Subject 5:** Yeah, possibly, but you do feel that you have to make a response, that it's your duty to make a response to the system because. **Interviewer:** Is it because your in an experimental setting or is it the system you are talking about the system. **Subject 5:** No, I think it's the system.

Here the learner has an external model of the system (RM system, which we will describe in Section 6.7) as a figure of authority that was pushing subject 5 to make a full response. This pressurised subject 5 (internal model of RM affect in terms of emotion). This extract demonstrates how the different types of RM may be related. (Note that '.....' in Table 3 denotes dialogue that the transcriber could not understand.)

6.2 Attention

Attention involves the sustaining of interest and selectivity in learning (Ashman and Conway, 1997). One consequence of RM attention could be, for example, that if learner A has a model of learner B that suggests B is not paying attention, then A may adopt the tutor's role. Table 4 shows a small example of RM attention from Session 5. Subject 9 is doing a bit of tutoring here. Subject 9

seems to have realised that Subject 10 is not showing enough interest. Subject 9 is taking on the role of the tutor here by asking a question like: "what would we expect?" (Note that '=[' denotes that the utterances above and below it start at the same time, '//' indicates an overlap, which is shown on the next line). It is interesting to note that subject 9 is in fact mimicking the kind of interventions that MetaMuse has been making (i.e. asking open ended questions).

Table 4: Example of RM attention

Utterance
Subject 10: Okay. Subject 9: So =[Subject 10: So Subject 9: Umm, okay, what // would we expect? Subject 10: What would we expect umm? Subject 10: It would repeat the thing.

Modelling another agent's attention can have other positive dimensions. In Session 6 we seemed to have an uncooperative session. However, during Session 6, subject 11 made at least two separate comments that indicated that some vicarious learning (McKendree et al., 1998) had took place (where an agent learns by observing another agent's learning activities). For example, at one point in the interactions subject 11 articulated what may have been an important self-realisation: namely that perhaps he had jumped in too quickly and did not read some of the help information provided by MetaMuse (i.e. subject 11 was being too selective). Subject 11 seemed to achieve this realisation by referring back to his past actions and by sitting back and watching subject 12 take his turn. Subject 11 even diagnoses himself as being a habitual offender in this respect.

6.3 Creative intention

In a cooperative context, creative intention can be characterised by the question: are we in agreement about the creative artefact that we have in mind, or not? This dimension of cooperative problem-solving can potentially have a big impact on the creative product. With a score of 17 occasions (Table 2) when RM Creative intention was observed in the sessions and the interview data, this RM dimension is the second highest scoring RM dimension. Working with another person to be creative is problematic. Some very positive interactions were observed in most of the sessions. However, cooperative problem-solving did not seem to go too well in Session 6. Interview data, for subject 11, that was obtained following Session 6 showed that he appeared to believe that he and his partner should work alone, that they should stick to their own ideas. Furthermore, in the interview following Session 3, subject 6 had clearly built up an image of the other learner's creative intentions, but expressed the view that she felt that she was being too influenced subject 5's ideas: "I think I was perhaps influenced by the way he was thinking".

6.4 Experiment

By RM experiment we mean that a learner may build up a model of what he or she thinks the purpose of the experiment is. In the interview following Session 1, subject 2 made the comments shown in Table 5 with respect to his interactions with the computer. Interestingly, Subject 2 is modelling the computer and the experiment. He is using his RM experiment as the basis of a critique of the experiment (in fact the objectives were perhaps new to him — i.e. making predictions — rather than being unclear; also the terms he is referring to would make sense to many musician).

Table 5: Example of RM experiment

Utterance
Subject 2: I would think something was talking to me if it respond in a more real manner ... And as I suppose part of my experience with this, and this is also my experience with computers, is trying to think what's wrong with it and what's right with it as a piece of software. As well as what we were trying to do. I was trying to do, I think I was like doing experiments on the experiments [LAUGHTER FROM SUBJECT 2] **Interviewer:** You were looking down on the experiment, is that what you're saying, and thinking about what was going on here. **Subject 2:** That's right yeah. I think the umm the objective wasn't quite clear and the terms weren't very clear. And I mean if I was an actual musician errm maybe these terms would have made sense to a musician I was guessing whether that was supposed to be the case as well.

6.5 Knowledge level

Knowledge can be declarative, procedural or entrenched. In Table 6 we see subject 1 declaring his own knowledge

level and asking a question to ascertain subject 2's knowledge level (in general terms). Furthermore, in the interview following Session 1, subject 2 suggested that subject 1's musical knowledge may have been limiting subject 1's openness to new ideas. This is what we are calling entrenched knowledge: where an agent seems set in one way of seeing knowledge, they do not seem to be able to break out and see knowledge from another viewpoint. Knowledge entrenchment may limit an agent's ability to problem-seek.

Table 6: Example of RM knowledge level

Utterance
Subject 1: Then it would be back [ADDS 0 TO LIST] again so I know what that's going to sound like because I'm a musician myself, but are you, do you know what it's going to sound like. **Subject 2:** No not really, I have a rough idea but it's very rough I think. Maybe you can help me. [LAUGHTER FROM BOTH SUBJECTS] See whether we hit or miss?

6.6 Social status

This is where a model is built up of the social status of another learner. A good example was provided by session 2, subject 4 made the following comments:

> "Well to be honest I thought she [subject 3] might have been a new Ph.D. student so I would have thought she would have been (more able) in coming forward. But when we were outside [following the session] and she said she's on the second year of her degree I thought, maybe she thought I was going on and on and on and it would be really rude to interrupt me. I thought she was shy at the time ...".

Subject 3 was aware that subject 4 was a member of academic staff. However, for the duration of the session subject 4 did not know that subject 3 was an undergraduate (although she may have had her suspicions; subject 4 thought subject 3 may have been a "new Ph.D. student"). The above modelling of subject 3 by subject 4 (which my also be related to affect) had a bearing on the manner in which cooperation took place (subject 4 decides not to say something at one point). This gives a good example of the care and consideration that may come into play when cooperating partners realise that there is a possible imbalance in social status between them.

6.7 System

This is where learners build up a model of the computer-based system that they are using. Users are likely, in some sense, to try to elicit designers' models during learning. Bullock et al. (1982) refer to this as the 'mechanism principle'. The principle claims that causal attribution is more plausible if there is a mechanism that could mediate the causal connection between two events. Empirical study by Pazzani (1987) suggests that this strongly influences human reasoning. A machine offers the user a chance to infer causal and procedural connections through implicit representation of dialogue.

With a score of 22 occasions (Table 2) when RM System was observed in the sessions and the interview data, this RM dimension is the highest scoring RM dimension. An interesting example of this RM dimension was given above in Table 3. The system (RM system) was seen as a figure of authority that was pushing subject 5 to make a full response. This induces some small pressure to answer in subject 5.

7 Summary

The taxonomy of reciprocal model types provides an analysis framework for describing the influence of third party intervention (e.g. the agent) on cognition during problem-solving tasks. Evidence suggests that reciprocal models have a considerable influence on user behaviour. Any attempt by a pedagogical agent to engage a user in a meta-dialogue, where the user problem-solving behaviour in effect becomes the 'conversational object', should take into account the fact that the user's model of the system-in-operation will be mediated through a model of the agent's intentions. The taxonomy of reciprocal models presented above represent a basis upon which the design of our pedagogical agent can be improved so that it is better able to engage in such meta-dialogues.

8 Conclusions and future work

In this paper we have described MetaMuse, a pedagogical agent that was designed to 'structure interactions' and hence facilitate problem-solving dialogues. We then went on to describe the results from a detailed analysis of the interactions that took place between cooperating students when engaged with MetaMuse. The main empirical results presented in this paper was the seven types of recip-

rocal models (RM) which revealed that RM played an important part in shaping learner interaction and cognition. This result give us a much more detailed picture of the interactive means by which learning agents engage in cooperative problem-solving. Future work will focus on making use of our empirical results in a re-implementation of MetaMuse with the aim of enabling our pedagogical agent to be better able to support cooperative problem-solving.

Because computational resources are finite, we need to target future areas of re-implementation of MetaMuse to areas that will maximise learner cooperation. Consequently, a focus will be placed on detecting the above occurrences of RM (e.g. social status) that lead to 'limited cooperation' and attempting to improve the situation. Ndiaye and Jameson (1994) address the issue of making their natural language processing system, called PRACMA, 'transmutable', i.e. where the system is able to take on either of two possible roles in a dialogue. Transmutability would, the authors claim, enhance the system's ability to anticipate and interpret a dialogue partner's reasoning and behaviour. PRACMA models non-cooperative dialogues between a buyer and a seller and includes a module that makes use of Bayesian meta-networks for reasoning about the dialogue partner's beliefs and evaluations. Clearly, such an approach would be useful because, as we saw above, a student may engage in 'limited cooperation'. A priority in the re-implementation of MetaMuse will therefore be given to implementing Bayesian meta-networks that are capable anticipating and interpreting limited cooperation. We anticipate that a re-implementation of MetaMuse along these lines will put our pedagogical agent in a better position to support learner cooperative problem-solving.

Acknowledgements

MetaMuse was developed as part of the Cook's Ph.D. work (awarded 1998), which was carried out at The Open University (UK) under the supervision of Michael Baker and Simon Holland. The initial evaluation of MetaMuse and the transcription of the dialogue data were done with the help of Matt Smith. Our thanks to the anonymous reviewers for their comments.

References

Ashman, A. F. and Conway, R. N. F. *An Introduction to Cognitive Education.* London: Routledge, 1997.

Bullock, M. Gelmen, R. and Baillargeon, R. *The Development of Causal Reasoning,* In The Developmental Psychology of Time, Friedman, W. (Ed.) New York: Academic Press, 1982.

Cook. J. Mentoring, Metacognition and Music: Interaction Analyses and Implications for Intelligent Learning Environments. *International Journal of Artificial Intelligence in Education,* 9, 45-87, 1998.

Cook, J. MetaMuse: A Teaching Agent for Supporting Musical Problem-Seeking and Creative Reflection. *Proceedings of AISB'99 Symposium on Musical Creativity,* pp. 89-95. Published by The Society for the Study of Artificial Intelligence and Simulation of Behaviour, 1999.

Cook, J. and Smith, M. Moving towards more effective musical teaching agents. *Proceedings of AISB'99 Symposium on Musical Creativity,* pp. 96-102. Published by The Society for the Study of Artificial Intelligence and Simulation of Behaviour, 1999.

Ericsson, K. A. and Simon, H. A. *Protocol Analysis (Revised Edition).* Cambridge, Massachusetts: MIT Press, 1993.

Lipman, M. *Thinking in Education.* New York: Cambridge University Press, 1991.

McKendree, J., Stenning, K., Mayes, T., Lee, J., and Cox, R. Why Observing a Dialogue may Benefit Learning. *Journal of Computer Assisted Learning,* 14(2), 110-119, 1998.

Ndiaye, A. and Jameson, A. Supporting Flexibility and Transmutability: Multi-Agent Processing and Role-Switching in a Pragmatically Oriented Dialog System. In Jorrand, P. and Sgurev, V. (Eds.), *Proceedings of the Sixth International Conference on Artificial Intelligence: Methodology, Systems, Applications* (pp. 381-390). Singapore: World Scientific Publishing, 1994.

Pazzani, M. Inducing Causal and Social Theories: A Prerequisite for Explanation-based Learning, *Proceedings of Fourth International Machine Learning Workshop,* Irvine CA, 230-241, 1987.

From individual to distributed minds in creative design: the re-representation hypothesis revisited

Alberto Faro and Daniela Giordano
Istituto di Informatica e Telecomunicazioni, University of Catania
viale A Doria, 6, 95125, Catania, Italy
afaro, dgiordan@iit.unict.it

Abstract

The paper recasts the Re-Representation Hypothesis (RRH) in a social design context that allows the designers to leverage on the collective experience of the community they belong to, taking as an example an evolving shared design memory which has been implemented to support the students of an Information Systems (IS) design course. The paper not only discusses how the cognitive value of the re-representation hypothesis may be preserved even in a social context, but also demonstrates how a distributed memory embedding this cognitive process can be built and used for both supporting design solutions by adapting precedents and generating novel knowledge schemas (ontological and representational abstractions) that become an asset for the community.

1 Introduction

Available models of the design process have been developed following disparate approaches, diversified along dimensions such as: the collaborative modalities by which the design is carried out (e.g., individual, collaborative design, participatory design, social design); the adopted constructivistic approach (e.g., objects, functions, activities), or the cognitive processes involved (e.g., stepwise refinement, analogical reasoning, metaphorical blendings). In the paper we are interested in the relationship between cognitive processes in design and the representations used to describe the system specifications, e.g., drawings and denotational systems, or notational, non-notational and discoursive representations (Willats, 1997; Goel, 1995). This relationship has received great attention to explain, respectively, the exploration of relevant concepts in the preliminary phase of design (see the role of ill-structured representations in Goel (1995)), and in detailing the chosen concept until the final design is achieved (see the role of the re-representations in (Karmiloff-Smith, 1993)). In particular, the Re-Representation Hypothesis (RRH) is currently accepted as the individual cognitive process (or cognitive strategy) that guides the production of a new solution starting from relevant precedents (Oxman, 1997). RRH involves a continuous re-coding of an initial tentative solution, constrained by the application of multiple representational abstraction levels (e.g., functional relations, topological grid systems, layering systems).

The paper recasts RRH in a social context that allows the designers to leverage on the collective experience of the community of designers they belong to. In particular, we consider the development of a design solution starting from relevant precedents drawn from an evolving shared design memory which has been implemented to support the students of an Information Systems (IS) design course. The designers have to contribute their design to the case base so that it incorporates precedents that may be at the cutting edge of the knowledge available to the community. This could be achieved, in the tradition of case-based reasoning (CBR) systems (Kolodner, 1993), by simply inserting the new cases according to the indexing scheme of the base. However, this way of proceeding implies that the individuals can access the new cases but do not have any explicit suggestion on how the creative process develops at the social level, neither about what representational abstraction levels, sometimes called "knowledge schemas", are being applied by the community. This could be achieved only by devising some means to access to the creative process emergent at the social level.

To do so we have to tackle two main issues: a) to extend the re-representation hypothesis (RRH), and b) to build a distributed memory that embodies this mecha-

nism. The main problem in extending RRH is that the available studies on this issue do not grant a natural extension of RRH to our target environment for several reasons:

a) the most explored design activities regard architectural or industrial design, with intermediate drawapproaches are usually downplayed, whereas cultural differences in design solutions have to be encouraged (especially in the IS field) so that the variety of available solutions may support innovation;

c) the intermediate representations are mainly sketches and drawings, whereas other visual techniques and multimedia environments can be used in design activities;

d) design is mostly investigated as an individual activity, though admitting that it can explicitely take advantage from external sources, e.g., from precedents (Kolodner, 1993 and Oxman, 1997), whereas designs are more productively developed by design teams operating in a much richer social context, not only when they deal with highly complex artifacts but also in common professional practice.

Thus the aim of the paper is twofold: a) to discuss how the cognitive value of the re-representations hypothesis may be preserved even in a wider design context, i.e. a social and distributed one, with a practical illustration in the field of Information Systems design, and b) to demonstrate how a distributed memory embedding this cognitive process can be built. The paper is organized as follows. Section 2 starts from the obsolescence of individualistic approaches to IS design - and of support environments based on this philosophy, to emphasize the importance of the social dimension and the key role of the shared design memory. Section 3 illustrates how re-representations productive for an evolving design may be supported by the distributed memory. Section 4 points out the special role played by aesthetics (all too often underrated in IS design) as a generalized language ensuring mutual intelligibility to the designers and also suggesting relevant dimensions to the implementors. Empirical evidence of the relevance of the revisited re-representation hypothesis in social contexts and of the role of the aesthetical dimensions in widening the exploration of the design spaces in the preliminary design and in suggesting non notational, but effective, design solutions carried through the final design phase is discussed in section 5.

2 Social approach to design and shared design memories

Modern theories of knowledge point out that knowledge is distributed in networks of social actors, resources and artifacts and that cognition is fundamentally situated and cultural process (Suchman, 1988; culturation, of being introduced to the modes of discourse of a particular community, of participating in a social process of making sense and negotiating understandings, mostly through narrative construction (Brown & Duguid, 1991). In taking seriously the idea that, in some sense, cognition can be «socially shared» or «distributed» one must consider, as pointed out by Cole (1996), that "much of what is meant by social in the notion of socially shared cognition is rather cultural, meaning by culture the full set of resources for constructing contexts, i.e., combinations of goals, tools shaped by the cultural past, and current circumstances, which constitute the *text* and *con-text* of behavior and the ways in which cognition can be said to be distributed in that context". This perspective leads us to consider that in understanding design cognition and creativity the community of practice and the social environment to which the designers belong must be part of the picture. This is obviously true of situations in which design, as a matter of professional practice is naturally carried out in teams (e.g. Bucciarelli (1993) account of designing engineers), but we would favour a view in which the social and distributed perspective is foundational and also accounts for activites that only in a very restricted sense can be considered to be carried out "individually".

A viable approach to support design activities in a social context is to resort to a "shared design memory" that is not simply a collection of design precedents but rather is a growing and evolving resource tracing the design experiences in which the designers of a specific community are engaged. Several approaches to support designers in the wider contexts of a team or of a whole corporation have been proposed, mostly devising forms of organizational memory to make design knowledge more resilient and more easily accessible, or to decrease the cost of re-considering design decision, e.g., by recording design rationale. In general, these interventions are aimed at increasing the cost-effectiveness of the design action and are not particularly concerned with the support of creativity. In particular, they do not favor new solutions by pointing out as many design aspects as possible that are relevant for the task at hand

and may come from both similar cases and apparently unrelated fields. On the contrary, the considerations advanced in this paper stem from the observation of the contents and their evolutionary dynamics in a shared design memory that was intentionally deployed to support learning and creative design, in the above sense, in a community of IS design students. Deployment of the system was accompanied by contextually establishing the culture of making use of the shared design memory and attempting design innovations.

Dealing with IS design has various consequences concerning the key problems of case representation and of indexing the contents of the memory, this latter typically being task-specific. In fact, the shared design memory must favour multiple traversal paths and support various search options. In IS design, ontologies concerned with the use-cases of the system are of paramount importance, thus one way of searching for relevant cases in the shared memory is according to the ontologies specification. This is demonstrated by the observed behaviour of the community of novice designers who often request not only precedents belonging to similar business activities but also precedents that contain similar use-cases, independently on the specific business activity. Thus, as an example, when designing the IS use-case of managing the enrolment to a university course it is not so important for the students to consult a precedent coming exactly the university domain, but they are fairly satisfied from consulting a precedent of similar kind, such as the one related to what happens in joining a gym. Of course it is also important that they are advised todeal with the commonalities of the use-cases as general schemas to be adapted for their needs, while specific implementations (such as the use of Internet for registering from home) should be regarded as possible innovations.

For supporting an effective remembering or retrieval of precedents respectively from personal and shared memories and following also the underlying methodological approach to design, these specifications are couched in terms of a template based on categories (i.e., WHAT, WHO, HOW, WHY etc.) that discoursively structure the specification of the activity streams that must be supported by the information system. These descriptions (story-episodes templates) are among the key design representations that are contained in the shared memory. To supplement the template are also the diagrammatic specifications that formalize and substantiate the template whose narrative structure favors the construction of understanding. These are Data Flow Diagrams (DFDs) aiming at specifying user and system processes and their interactions, and Entity Relation Models (ERMs) aiming at describing and optimizing data structures. Also attached to the template are representations dealing with the dynamics of the system: Entity Life Histories (ELH) aiming at describing diagrammatically what processes at any given time may access the data and for what reason, and Automata or Petri Networks aiming at describing how users and system processes should concurrently behave for increasing as much as possible both effectiveness in user-system interactions and efficiency in system implementation.

One constructive assumption for the shared design memory (which corresponds to an implicit cognitive hypothesis) is that such a social resource is effective to the extent that the contributions are linked in a network of influences that trace how a new design has evolved from the already available precedents (Faro&Giordano, 1998). This is to say that the meaning of an ontology and the value of a particular design solution is evident in the context where the case is embedded. We assume that this way of structuring the content is more powerful than what could be accomplished with broad association links or with notions of similarity based on ad-hoc indexing scheme (e.g., as in issue-based systems or the problem-form-concept triple proposed in Oxman (1999)) with respect to the fundamental issue of detecting the unexpected and the emergent in the community. This can be accomplished by endowing the shared design memory with a mechanism of neural classification that operates on the space of the links (Faro&Giordano, 1998). In fact, unsupervised neural classification originates classes that can provide refined, reinforced or new ontological schemas based on the cases that are flexibly aggregated in each class. Examining such classes is an additional way of looking at the contents of the memory. This amounts to provide a socially emerging supplementary indexing scheme that reflects the issues that are of importance to the community. As an example, fig.1 shows a 3D space of this N-dimensional topology where small nodes represent use-cases whose position Px, Py, Pz in that space represents how much (from 0 to 1) a case P belongs to the classes Cx, Cy and Cz, respectively dealing with Medical Visit, Service Advertising, Course Enrolment. The interpretation of this topological representation is as follows: a) the more the cases belong to only one axis and the more they carry out only one aspect, b) a case is influenced by another case if it is connected to the other by an arc whose arrow is in the direction of the influential one, and c) the arcs are characterized by a pair weight-comment to describe how much and why the case at one end of the arc was influenced by the one at the other end.

Thus, with respect to fig.1, if a designer is interested in specifying the enrolment to a gym she\he could limit her\himself to consulting only the cases belonging to the kernel of the "Course Enrolment" ontology (i.e., the ones contained in the bigger black node drawn in axis

x). However, she\he may follow a more creative strategy by analyzing the aspects other cases suggest to be relevant too, i.e., the medical visit to verify the client condition before accepting the enrolment request and the advertising action to increase the number of enrolments. Of course if no case exists like cases P1 and P2 in fig.1, then analysis of new aspects depends on the culture and background knowledge of the designers. But it suffices that one member of the community inserts such a case for making the innovative aspects developed in the new case available to be considered by the community for increasing the effectiveness and efficiency of the already existing enrolment ontology.

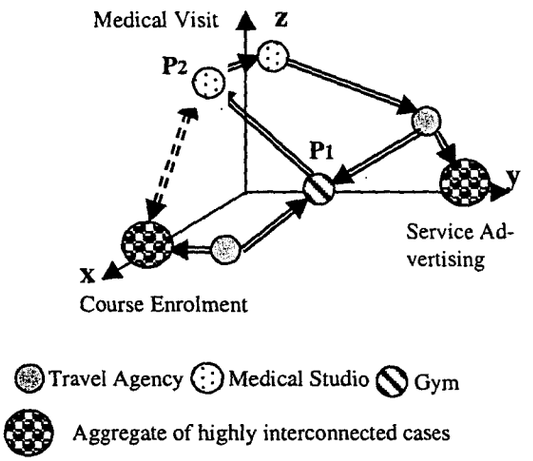


Fig.1 – Use cases arranged in a 3D ontological topology created by a neural agent in the shared memory.

3 Re-representation revisited

In this section we focus on design creativity, i.e., the cognitive processes we want to support for creative design to take place. To this aim we assume, as a first approximation, that at both the team and the social level the design process develops following four main subsequent phases, as proposed in (Goel,1995) for explaining individual design activities, i.e.: i) problem structuring, ii) preliminary design, iii) revision and iv) detailed design. With respect to this model, creative activities occur as "lateral transformations" in search of an initial tentative drawing and relevant aspects, and "vertical transformation" of the drawing chosen as a starting one until the final solution is achieved. A comparison with the re-representation hyphotesis discussed by Oxman (1997) would suggest that Goel is actually claiming more than one cognitive mechanisms to explain creativity in design, since re-representation might be appropriate to explain how designers think in the detailed design phase, but not in the preliminary one, in which the representations taken into account are not obtained one from the other, as RRH requires. However, a closer look tells a different story, given that RRH implies that the designer needs to use some knowledge schema to produce a new representation from a previous one. Preliminary design seems to serve exactly to this aim, i.e., to select, or more generally, to discover useful schemas that can guide the RRH process.

Thus a generalized creative behavior might assume that not only the starting representation but also the knowledge schemas cannot be defined in advance. Rather, they should be situationally identified or activated, depending on the expertise level, among the ones already available or emergent from the individual designer's memory. Experimental evidence of this generalized hyphotesis can be found in both the etnographic studies from Goel (1999) and Oxman (1999), even if neither of these two authors enlarges his hypotesis to comprise the point of view of the other.

However, if RRH and generalized RRH had this formulation one couldn't avoid the criticism presented by Carassa & Tirassa (1994) in which RRH is seen as a strategy for creative behavior rather than a cognitive mechanism. In fact, all the above mentioned authors do not have illustrated neither hypothesized how this mechanism can be implemented in a cognitive architecture.

One could be tempted to discount the relevance of this criticism, but it is reasonable to think that the cognitive load in following a strategy (i.e., exploring personal design experience and expliciting personal knowledge schema) is greater than the one needed to do this according to some internal cognitive mechanism. Actually, this issue is at the outset of approaches that postulate modular connessionistic cognitive architectures aimed at better exploiting achieved representations vs. engaging in expensive computation (Clark & Thornton, 1997) or even in more radical positions that completely exclude the existence of representations in the brain, favouring an increasingly refined classification of the problem that allows one to "see" the solutions (Dreyfus, 1998).

For this reason in conceiving of the distributed memory of a community of designers we have always been interested in a memory structure that incorporates a generalized RRH or re-classification mechanism as a native re-representation facility to understand what aspects are relevant and how they can be couched in a novel solution. This has been obtained by the neural agent introduced in the previous section where the mentioned mechanisms work as follows:

1) Exploration of the memory in search of relevant schemas (design aspects or concepts) to be taken into account during the design process may be easily done by first selecting (e.g., by a lexical query) a relevant use story S in the memory space, then by exploring the class Cs to which this story belongs to and finally by exploring the use-stories belonging to the neighboring classes of class Cs or the use stories belonging to the intersections between class Cs and the neighboring ones. With respect to fig.1 this means that to start the re-representation process of the "course enrolment in a gym" it is preliminarly necessary to choose what ontologies should be taken into account in addition to the ones believed fundamental for the case at hand. It is at this stage that a creative designer could decide to incorporate in her\his use-case the medical visit and the advertising. Of course if the future cases will reinforce this choice, the ontology will evolve towards a new schema that stably incorporates the above new aspects, otherwise the proposal could disappear or be accepted simply as a significant variant.

2) Exploration of the memory reclassified or "re-framed", respectively, according to the following simple mechanisms:

a) neural reclassification of the part of memory in the surround of the initial relevant use stories;

b) neural reclassification of the memory after having relaxed the links interconnecting the nodes of the class Cs.

Both these operations could let emerge new classes obscured by the original classification. Re-classifying the cases of a class will let further internal aspects emerge. On the contrary, decreasing the weights of the internal links will let potential ontologies emerge. As an example, fig.2a shows a possible result of re-classifying the cases belonging to the axis x of fig.1 by taking into account only their interconnections with the other cases of the 3D space. As a consequence, class Cx will be subdivided in some classes such as Cx1 more linked to Class Cz, and Cx2 to class Cy. These sub-classes deal with aspects of the original one. Fig.2b shows what happens if the weights of the internal links of classes Cx1 and Cx2 are furtherly decreased: some cases (class C'x) will disappear, whereas the other cases will be subdivided in two classes: C'x1 and C'x2 that will let potential ontologies emerge.

3) Adapting a solution of the memory (i.e. a relevant precedent) by taking into account: the re-representation trace that has detemined the evolution of the precedents and the aspects emerged in the exploration phase. The linkage of the new solution to the ones that have been re-used in the creation process increases the re-representation path and may reinforce the schema the designer has used in deriving the new solution from the existing relevant ones.

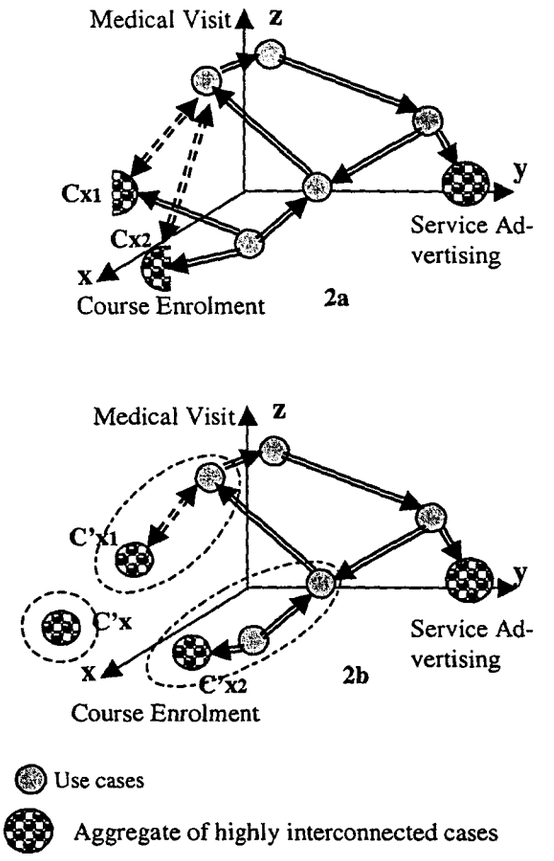


Fig.2 Re-framing case topology

Let us note that the first two points above are in accordance with Goel's hypothesis dealing with the necessity of selecting the knowledge schemas preliminarly to start the design process by using non notational diagrams such as sketching, with the difference that we assume that designers may take into account new schemas during the design process too. On the contrary, the last point (3) is partially in accordance with Oxman's model of the design process since in our model the re-representation process may result not only in a new solution but also in reinforcing a new schema.

This means that the knowledge schemas are not fixed but that they socially evolve, whereas re-representation at both the individual and shared memory levels does not differ significatively since in both cases it is a path towards a final solution, with the difference that the final solution for the individuals is an intermediate solution at the social level. In fact re-representation in distributed memory can be interpreted as a path towards some stable ontology (i.e. the final solution that social

supports an effective representation of concurrent processes. An example of style linked to a specific ontology and suggested by the shared memory is the one consisting of videos, organizational charts and photos that characterize the introductive section devoted to describe the main scenarios of the office automation support system. An example of representations initially considered as an unusual combination is the colored global data flow diagrams where globality means the representation in the same diagram of all the system's processes and their interactions, whereas coloring is a technique created by the community with the aim of differentiating the interactions according to the use-case to which they belong to. An example of a completely new representational technique is the actor-story matrix that aims at verifying that an actor should be present at least in a story and, viceversa, that a story should be enacted by some actor. Some of these new representational styles will be furtherly discussed in the next section.

5 Empirical evidence

This section briefly presents, with the help of the diagrams shown in fig.3 and fig.4, some empirical evidence that justifies why the above mentioned findings modify the models proposed by Goel and Oxman in two main senses:

a) initial exploration regards both ontological and aesthetical schemas; these schemas are partially owned by the designers but may also be discovered during the exploration of both the informal and artificial distributed memory. IS design doesn't evolve according to a linear sequence of phases neither as the feedback evolution proposed by Oxman, but according to an impredictable interleaving of exploration and re-representation.

b) the necessity of suggesting a form to the implementors and of adjusting the user requests to the design notation determines mixed represent-ations starting from the structuring of the problem until the final solution is achieved. In particular, a certain degree of notational representation is always present, whereas discoursive representations are mainly present in the early design, and drawings in both the preliminary and detailed phases of the design.

Fig.3b shows a DFD dealing with the IS of a gym that has been inspired by a former DFD shown in fig.3a dealing with the IS of an airport. Influences mainly concern the representational dimensions as clearly emerges from the comparison of the two diagrams. Interestingly, some aspects dealing with managing the clients have been "imported" from the precedent to the new project too.

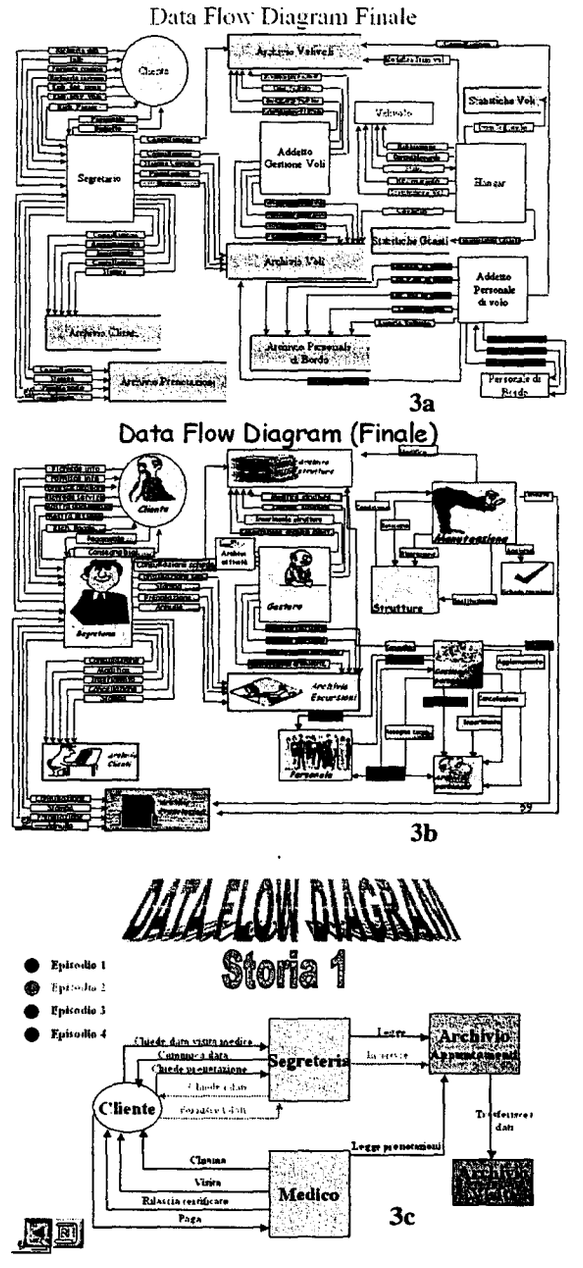


Fig. 3 – Some re-representations of former cases

Both projects in fig.3b and fig.3c deal with the same subject and have been produced in the same period. Of course they share a common kernel related to managing the clients and the gym courses. However, the case shown in fig.3b deals also with the gym maintenance, whereas the other specifically treats another important aspect concerning the medical visit for accepting the client in the gym. They are a good example of re-representing a prior ontology by adding two different aspects that may become stable points in the future projects. From the representational point of view the

only similarity is that both diagrams point out the actions (and related roles) played by the identified processes in the use-episodes to which they take part.

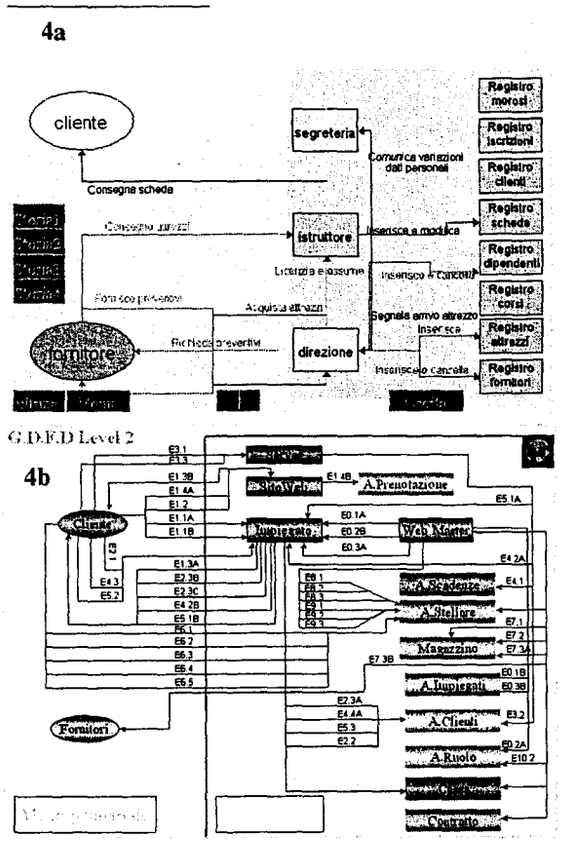


Fig.4 – Re-applying a representational schema

Fig.4a is another example of gym management. Let us note a completely different stylistic schema with respect to the one shown in fig.3b. This is because the design aims at pointing out especially the data base organization. In this respect the latter DFD is very close to the style of the diagram shown in fig.4b aiming at structuring the data base of an astrophysical center which was used as a precedent.

6 Concluding remarks

The above DFDs are in a certain sense in the middle of the design process. They represent the point from which the use of discoursive part is abandoned in favor of diagrammatic representations enriched by colors, iconography and animation. The textual part, if any, concerns only some comments to better animate the simulation (e.g., the Petri Network) or to explain the rational of some choices concerning the system structure. This result is less surprising if one thinks that relevant examples exist in the art in which both denotational and drawing systems coexist, e.g., in Van Gogh's Jardin de fleurs (1888) as noted in Willats (1997), to express both the inner contents and the observer point of view.

References

Brown J, & Duguid P. Organizational learning and communities of practice, *Organization Science*, 2,1, 40-57, 1991

Bucciarelli, L. *Designing engineers*, MIT Press, 1994

Carassa, A., and Tirassa M. Representational redescription and cognitive architectures, *Behavioral and Brain Sciences*. Vol.17. pp. 711-712, 1994

Clark,A. & Thornton, C. Trading spaces: Computation, representation, and the limits on uninformed learning", *Behavioral and Brain Sciences*, 20:1, 57-67, 1997

Cole, M. Conclusions in *Perspectives in socially shared cognition*, L. Erlbaum Associates, 1996

Dreyfus H.L. Intelligence without representation, *Cognitive Science Humanities and the Arts*, url: http:\\www.hfac.uh.edu\cogsci\index.html 1998

Faro A., & Giordano D. Concept formation from design cases: why reusing experience and why not. *Knowledge Based Systems Journal* N.7-8, Elsevier, 1998

Faro A., & Giordano D. Ontology, aesthetics and creativity at the crossroad in information systems design. *ACM Proc. on Creativity and Cognition*, Loughborough, UK 1999

Goel V. *Sketches of thought*, MIT Press 1995

Goel V. Cognitive role of ill structures representations in preliminary design, *Proceedings on Visual spatial reasoning in design*, MIT, Boston, 1999

Hutchins, E. *Cognition in the wild*, MIT Press, 1996

Karmiloff-Smith A. Constraints on representational change: evidence from children's drawing, *Cognition* vol.34 pp 57-83, 1993

Kolodner, J. *Case Based Reasoning* San Mateo, CA: Morgan Kaufmann, 1993

Lave, J. *Cognition in practice.* Cambridge University Press, 1988

Oxman R. Design by re-representation: a model of visual reasoning in design, *Design Studies*, vol.18 N.4, Elsevier, 1997

Willats J. *Art and representation*, Princeton University Press, 1997

much on the quality of the concept maps involved, this being the motivation leading to the development of *Clouds* (Pereira, 2000), a module now integrated in *Dr. Divago*, aiming at helping the user in building her own concept maps by means of machine learning techniques.

In the present work, within *Dr. Divago*, we want to extend the generation of cross-domain links such as those based on the metaphor theory of Veale and Keane (1993), to cope with the transfer of operational/procedural knowledge from one domain to another. In such cases, particularly when negative information is bought into the arena, inconsistency is a reality that has to be dealt with, as shown by the following example:

Example 1 *Consider the rule* "objects aren't big when they are in the background", *extracted from* Clouds *in the domain of Visual Arts, represented in the clausal form as:*

$$not\ big(X) \leftarrow object(X), background(X). \quad (1)$$

which, if we were to map it to the music domain according to the following translation:

$$big \rightarrow long \qquad object \rightarrow motif$$
$$background \rightarrow accompaniment$$

would correspond to:

$$not\ long(X) \leftarrow motif(X), accompaniment(X). \quad (2)$$

This can easily be proven inconsistent, at least for some instances (it is not difficult to find accompaniment motifs *that are* long, *eg. isometric motets from ars nova).* ◇

Each metaphorical mapping will lead to the transfer of a set of rules from the *Vehicle* to the *Tenor* of which: some are redundant because equivalent rules already exist there; some, of particular interest to the creative process, yield new knowledge to the *Tenor*; some are contradictory with the knowledge already present in the *Tenor*. Although contradiction is not necessarily evil, this being more evident in creative processes, we nevertheless have to detect and deal with it.

The need to detect and deal with contradiction in knowledge transfer lead us to consider the recent paradigm of *Dynamic Logic Programming* (Alferes et al., 1998, 2000) as a formal, while at the same time intuitive, vehicle to a solution.

The paradigm of *Dynamic Logic Programming (DLP)*, supported by the notion of *Logic Program Updates* (Leite, 1997; Leite and Pereira, 1997, 1998; Alferes et al., 1998, 2000), is simple and quite fundamental. Suppose that we are given a set of theories (encoded as generalized logic programs) representing different states of the world. Different states may represent different time periods or different sets of priorities or, in our case, different domains. Consequently, the individual theories contain mutually contradictory as well as overlapping information. The role of *Logic Program Updates* is to use the mutual relationships existing between different states to precisely determine the *declarative* as well as the *procedural* semantics of the combined theory, composed of all individual theories. Cross-domain inconsistencies are detected and solved due to the application of the principle of inertia to the individual rules of the theories. This principle of inertia, applied to the individual rules of a theory, states that a rule from the initial knowledge should only persist in time, after an update, if it does not lead to a contradiction by means of a new rule. In our case, the rules from the *Vehicle*, after the metaphorical mapping, should only persist if they do not lead to a contradiction by means of a rule from the *Tenor*.

Going back to the example above, let us suppose that we learn, in the music domain, the following rule:

$$long(X) \leftarrow motif(X), accompaniment(X), \\ isometric\_motet\_element(X). \quad (3)$$

stating that *"the accompaniment element of an isometric motet is long"*[1]. If we were simply to consider the union of both rules, we would obtain a contradiction for all *accompaniment elements of an isometric motet*. If we were to simply consider rule 3, we would loose valuable information from rule 2 concerning *accompaniment motifs not belonging to isometric motets*. Our goal is to be able to conclude that *"accompaniment elements of an isometric motet are long"*, and that *"accompaniment elements of non-isometric motets aren't long"*. That is, we would like to use rule 2 for those instances that do not generate a contradiction by means of rule 3. This is precisely the behavior of *Logic Program Updates*: to exert inertia on those rules (in this case obtained by the mapping) that are not contradicted by rules from the new domain. In Example 3, we will return to this problem and show how to obtain the desired result.

The non-monotonic behavior of DLP, together with its modular characteristic shows to be quite important when dealing with metaphors.

This paper presents a symbiosis of the two above mentioned theories (*DLP* and *Metaphor*) towards the goal of *Computational Creativity*. It is being implemented as part of the above mentioned *Dr. Divago* (Pereira, 1998), where interesting results are to be expected.

Enjoying the advantages of employing a theoretically sound formalism to the problem of *Metaphorical Reasoning*, DLP has also been implemented as a meta-interpreter (DLP System, 1998) running under the XSB System (1999), allowing for not only theoretical but also practical reasoning.

The paper is structured in the following way: in Sect.2 we elaborate on *Metaphor Theories*; in Sect.3 we introduce the reader to the notions of *Logic Program Updates* and DLP; in Sect.4 we set forth the notion of *Metaphorical Reasoning* by means of DLP; in Sect.5 we draw some properties and illustrate with examples; in Sect.6 we conclude and give hints about future developments.

[1] Also known as Talia

2 Metaphor Theory

Metaphor is a constantly used resource of communication, be it as an embellishment for discourse or as a necessary device to assess concepts unexplainable in other ways. Since in any Natural Language, lexicon is not static or exhaustive, Metaphor has a constructive role in the evolution of communication. Words or expressions like "software", "e-mail" or "search engine" are recent examples of metaphors that are becoming entries in our dictionaries. Furthermore, some researchers advocate that Metaphor is a central device for Learning (Winston, 1980) and is an irreducible and irreplaceable function at the basis of our creative faculties (Richards, 1936; Black, 1962).

There seems to be general agreement that metaphor involves two objects or situations and some kind of transference from an object or situation to the other. One object is referred to as the *tenor* or the target domain, and the other object as the *vehicle* or the source domain. For example, the expression "Your claims are *indefensible*" follows the metaphor "Argument is War"(Lakoff and Johnson, 1980), in which an argument (the tenor) is partially defined, understood, performed and talked about in terms of a war (the *vehicle*). Here, we can observe the *Systematicity* of Metaphor: it is not limited to a single, static, association between two concepts (Argument and War). Instead, other associations emerge dynamically as one explores further in both domains, i.e., "Argument is War" is the underlying metaphor for "He *attacked every weak point* in my argument", "His criticisms were *right on target*", "I *demolished* his argument", "If you use that *strategy*, he'll *wipe you out*", and others that appear in everyday speech.

An interesting and subtle property of a metaphorical interpretation is that it is directional, i.e., each domain or object has a different role and its interchange, although possibly yielding an equally valuable metaphor, will not lead to the same meaning. For example, if we have the metaphor "War is Argument", things turn quite different. It is somewhat unconventional to talk about a war as being an argument (then, it would be natural to say "the general *exposed his claims* quite aggressively" or "thousands of soldiers perished from that *discussion*") because this metaphor is not so deeply rooted in our common sense reasoning.

While it is natural of metaphor to be associated with creative thought and freedom of association, it is constrained by deep rules of coherency. Although it is not expected, in a metaphor, to find a mapping for every single concept in a domain, those that are mapped should be coherent among themselves (this phenomenon is called Local Coherency in Indurkhya (1992)). Some research on metaphor interpretation consist precisely in finding the largest mapping function that avoids inconsistencies. One example is Veale and Keane's metaphor interpretation framework, called *Sapper* (Veale and Keane, 1993). It uses a hybrid model of semantic memory that consists of a connectionist structure in which each unit (or node) or fixed cluster of units is assigned to a concept, and an activation-carrying inter-unit linkage is assigned to each inter-concept relation. Also known as a *localist* network, this organization receives all domain knowledge Sapper uses to interpret metaphor. It applies 3 operational principles:

1. Metaphor is a means of learning new conceptual structure by linking existing diverse schemata in novel ways. This linkage of domains is achieved by augmenting the network with *conceptual bridges* that link the *tenor* and *vehicle* schemata of the metaphor (Veale, 1995).
2. Metaphor comprehension involves *explicit* structural changes to the semantic memory network; these changes, essentially the conceptual bridges of (1) above, are explicit inasmuch as they are recognizably the trace residue of a novel metaphor, and as such may be built upon (elaborated) at a later time (Veale, 1995).
3. A metaphor is a dynamic conceptual *agency*, which may continue to grow as more conceptual structure is acquired regarding either the *tenor* and the *vehicle* domains.

In Sapper, metaphor interpretation takes two major cyclic steps:

1. In the symbolic mode of processing, it searches for potentially inter-concept relations between the two different domains (*tenor* and *vehicle*). These relations, to which Veale calls *dormant bridges*, are obtained from the application of the following two rules:
   - The triangulation rule: "Whenever two concepts share an association with a third concept, this association provides for a plausible dormant bridge".
   - The squaring rule (second order similarity): If two concepts share an association with other two concepts that are connected by an awaken bridge (as described below), this association provides for a plausible dormant bridge.
2. The conceptual bridge awakening phase is performed in the connectionist mode of processing, in which dormant bridges, as laid down in the symbolic mode, are recognized to represent *domain crossover points* between the *tenor* and *vehicle* schemata, and are thus *awakened* or *burnt in*.

We are adopting these ideas to develop our cross-domain links generator, which allows to obtain metaphorical program mappings, described later in this paper.

Other works related to metaphor are also meaningful, such as Barnden (1997); Gentner et al. (1989); Martin (1990).

3 Dynamic Logic Programming

The idea of *dynamic program updates*, inspired by Leite (1997), is simple and quite fundamental. Suppose that we are given a set of program modules P_s, indexed by different states of the world s. Each program P_s contains some knowledge that is supposed to be true at the state s. Different states may represent different time periods or different sets of priorities or, in our case, different domains. Consequently, the individual program modules may contain mutually contradictory as well as overlapping information. The role of the *dynamic program update* $\bigoplus \{P_s : s \in S\}$ is to use the mutual relationships existing between different states (and specified in the form of the ordering relation) to precisely determine, at any given state s, the *declarative* as well as the *procedural* semantics of the combined program, composed of all modules.

Consequently, the notion of a *dynamic program update* supports the important paradigm of *dynamic logic programming*. Given individual and largely *independent* program modules P_s describing our knowledge at different states of the world (for example, the knowledge acquired at different times), the *dynamic program update* $\bigoplus \{P_s : s \in S\}$ specifies the exact meaning of the union of these programs. Dynamic programming significantly facilitates modularization of logic programming and, thus, modularization of non-monotonic reasoning as a whole.

In this section we start by recalling the definition of the so called generalized logic programs and their stable semantics (Alferes et al., 1998) which extend the stable semantics of normal logic programs (Gelfond and Lifschitz, 1988). We then recall the definition and semantic characterization of the update of a generalized logic program by means of another such logic program $P_1 \oplus P_2$.

Since we have that the notion of *dynamic program update* $\bigoplus\{P_s : s \in S\}$ over an ordered set $\mathcal{P} = \{P_s : s \in S\}$ of logic programs is a generalization of the notion of single program updates $P_1 \oplus P_2$, throughout the remainder of the paper, we will restrict ourselves, without loss of generality, to the case of single program updates $P \oplus U$.

For a formal and detailed presentation of *dynamic program update* and *dynamic logic programming*, the reader is referred to Alferes et al. (1998, 2000).

3.1 Generalized Logic Programs and their Stable Models

In order to represent *negative* information in logic programs and in their updates, we need more general logic programs that allow default negation *not A* not only in premises of their clauses but also in their heads. We call such programs *generalized logic programs*.

It will be convenient to *syntactically* represent generalized logic programs as *propositional Horn theories*. In particular, we will represent default negation *not A* as a standard propositional variable (atom). Suppose that $\mathcal{K}$ is an arbitrary set of propositional variables whose names do not begin with a "*not*". By the propositional language $\mathcal{L}_\mathcal{K}$ *generated* by the set $\mathcal{K}$ we mean the language $\mathcal{L}$ whose set of propositional variables consists of:

$$\{A : A \in \mathcal{K}\} \cup \{not\ A : A \in \mathcal{K}\}.$$

Atoms $A \in \mathcal{K}$, are called *objective atoms* while the atoms *not A* are called *default atoms*. From the definition it follows that the two sets are disjoint.

By a *generalized logic program* P in the language $\mathcal{L}_\mathcal{K}$ we mean a finite or infinite set of propositional Horn clauses of the form:

$$L \leftarrow L_1, \ldots, L_n$$

where L and L_i are atoms from $\mathcal{L}_\mathcal{K}$. If all the atoms L appearing in heads of clauses of P are objective atoms, then we say that the logic program P is *normal*. Consequently, from a syntactic standpoint, a logic program is simply viewed as a propositional Horn theory. However, its *semantics* significantly differs from the semantics of classical propositional theories and is determined by the class of stable models defined below.

By a (2-valued) *interpretation* M of $\mathcal{L}_\mathcal{K}$ we mean any set of atoms from $\mathcal{L}_\mathcal{K}$ that satisfies the condition that for any A in $\mathcal{K}$, precisely one of the atoms A or *not A* belongs to M. Given an interpretation M we define:

$$M^+ = \{A \in \mathcal{K} : A \in M\}$$

$$M^- = \{not\ A : not\ A \in M\} = \{not\ A : A \notin M\}.$$

Definition 1 (Stable models of generalized logic progs.) *We say that a (2-valued) interpretation M of $\mathcal{L}_\mathcal{K}$ is a stable model of a generalized logic program P if M is the least model of the Horn theory $P \cup M^-$:*

$$M = Least(P \cup M^-) \diamond$$

Following an established tradition, from now on we will often be omitting the default (negative) atoms when describing interpretations and models.

3.2 Program Updates

Suppose that $\mathcal{K}$ is an arbitrary set of propositional variables, and P and U are two generalized logic programs in the language $\mathcal{L} = \mathcal{L}_\mathcal{K}$. By $\widehat{\mathcal{K}}$ we denote the following superset of $\mathcal{K}$:

$$\widehat{\mathcal{K}} = \mathcal{K} \cup \{A^-,\ A_P,\ A_P^-,\ A_U,\ A_U^- : A \in \mathcal{K}\}.$$

This definition assumes that the original set $\mathcal{K}$ of propositional variables does not contain any of the newly added symbols of the form $A^-, A_P, A_P^-, A_U, A_U^-$ so that they are all disjoint sets of symbols. If $\mathcal{K}$ contains any such

symbols then they have to be *renamed* before the extension of $\mathcal{K}$ takes place. We denote by $\widehat{\mathcal{L}} = \mathcal{L}_{\widehat{\mathcal{K}}}$ the extension of the language $\mathcal{L} = \mathcal{L}_{\mathcal{K}}$ generated by $\widehat{\mathcal{K}}$.

Definition 2 (Program Updates) *Let P and U be generalized programs in the language $\mathcal{L}$. We call P the original program and U the updating program. By the update of P by U we mean the generalized logic program $P \oplus U$, which consists of the following clauses in the extended language $\widehat{\mathcal{L}}$:*

(RP) Rewritten original program clauses:

$$A_P \leftarrow B_1, \ldots, B_m, C_1^-, \ldots, C_n^- \qquad (4)$$
$$A_P^- \leftarrow B_1, \ldots, B_m, C_1^-, \ldots, C_n^- \qquad (5)$$

for any clause

$$A \leftarrow B_1, \ldots, B_m, \ not\, C_1, \ldots, \ not\, C_n$$

respectively

$$not\, A \leftarrow B_1, \ldots, B_m, \ not\, C_1, \ldots, \ not\, C_n$$

in the original program P. The rewritten clauses are obtained from the original ones by replacing atoms A (resp. not A) occurring in their heads by the atoms A_P (resp. A_P^-) and by replacing negative premises not C by C^-.

(RU) Rewritten updating program clauses:

$$A_U \leftarrow B_1, \ldots, B_m, C_1^-, \ldots, C_n^- \qquad (6)$$
$$A_U^- \leftarrow B_1, \ldots, B_m, C_1^-, \ldots, C_n^- \qquad (7)$$

for any clause

$$A \leftarrow B_1, \ldots, B_m, \ not\, C_1, \ldots, \ not\, C_n$$

respectively

$$not\, A \leftarrow B_1, \ldots, B_m, \ not\, C_1, \ldots, \ not\, C_n$$

in the updating program U. The rewritten clauses are obtained from the original ones by replacing atoms A (resp. not A) occurring in their heads by the atoms A_U (resp. A_U^-) and by replacing negative premises not C by C^-.

(UR) Update rules:

$$A \leftarrow A_U \qquad A^- \leftarrow A_U^- \qquad (8)$$

for all objective atoms $A \in \mathcal{K}$. The update rules state that an atom A must be true (resp. false) in $P \oplus U$ if it is true (resp. false) in the updating program U.

(IR) Inheritance rules:

$$A \leftarrow A_P, not\, A_U^- \qquad A^- \leftarrow A_P^-, not\, A_U \qquad (9)$$

for all objective atoms $A \in \mathcal{K}$. The inheritance rules say that an atom A (resp. A^-) in $P \oplus U$ is inherited (by inertia) from the original program P provided it is not rejected (i.e., forced to be false) by the updating program U. More precisely, an atom A is true (resp. false) in $P \oplus U$ if it is true (resp. false) in the original program P, provided it is not made false (resp. true) by the updating program U.

(DR) Default rules:

$$A^- \leftarrow not\, A_P, not\, A_U \qquad not\, A \leftarrow A^- \qquad (10)$$

for all objective atoms $A \in \mathcal{K}$. The first default rule states that an atom A in $P \oplus U$ is false if it is neither true in the original program P nor in the updating program U. The second says that if an atom is false then it can be assumed to be false by default. It ensures that A and A^- cannot both be true. $\diamond$

3.3 Semantic Characterization of Program Updates

Follows a semantic characterization of update programs $P \oplus U$ by describing their stable models. This characterization shows precisely how the semantics of the update program $P \oplus U$ depends on the syntax and semantics of the programs P and U.

Let P and U be *fixed* generalized logic programs in the language $\mathcal{L}$. Since the update program $P \oplus U$ is defined in the extended language $\widehat{\mathcal{L}}$, we begin by first showing how interpretations of the language $\mathcal{L}$ can be extended to interpretations of the extended language $\widehat{\mathcal{L}}$.

Definition 3 (Extended Interpretation) *For any interpretation M of $\mathcal{L}$ we denote by $\widehat{M}$ its extension to an interpretation of the extended language $\widehat{\mathcal{L}}$ defined, for any atom $A \in \mathcal{K}$, by the following rules:*

$$A^- \in \widehat{M} \quad iff \quad not\, A \in M$$
$$A_P \in \widehat{M} \quad iff \quad \exists A \leftarrow Body \in P \wedge M \models Body$$
$$A_P^- \in \widehat{M} \quad iff \quad \exists not\, A \leftarrow Body \in P \wedge M \models Body$$
$$A_U \in \widehat{M} \quad iff \quad \exists A \leftarrow Body \in U \wedge M \models Body$$
$$A_U^- \in \widehat{M} \quad iff \quad \exists not\, A \leftarrow Body \in U \wedge M \models Body. \diamond$$

We will also need the following definition:

Definition 4 *For any model M of the program U in the language $\mathcal{L}$ define:*

$$Defaults[M] = \{not\, A : M \models \neg Body, \forall (A \leftarrow Body) \in P \cup U\};$$

$$Rejected[M] = \{A \leftarrow Body \in P : \exists\, (not\, A \leftarrow Body') \in U \text{ and } M \models Body'\} \cup \{not\, A \leftarrow Body \in P : \exists\, (A \leftarrow Body') \in U \text{ and } M \models Body'\};$$ $\diamond$

The set $Defaults[M]$ contains default negations *not* A of all *unsupported* atoms A, i.e., atoms that have the property that the body of every clause from $P \cup U$ with the head A is false in M. Consequently, negation *not* A of these unsupported atoms A can be assumed by default. The set $Rejected[M] \subseteq P$ represents the set of clauses of the original program P that are *rejected* (or contradicted) by the update program U and the interpretation M.

Now we are able to describe the semantics of the update program $P \oplus U$ by providing a complete characterization of its stable models.

Theorem 1 (Stable models of update programs) *An interpretation N of the language $\widehat{\mathcal{L}} = \mathcal{L}_{\widehat{\mathcal{K}}}$ is a stable model of the update $P \oplus U$ if and only if N is the extension $N = \widehat{M}$ of a model M of U that satisfies the condition:*

$$M = Least(P \cup U - Rejected[M] \cup Defaults[M]) \diamond$$

4 Metaphorical Reasoning and DLP

In this section, we will show how the ideas behind *Dynamic Logic Programming* can be used in the context of *Metaphorical Reasoning*.

As we have seen before, a metaphorical framework can be seen as consisting of two theories (*tenor* and *vehicle*), defined in two different languages, together with a function mapping part of the language of the *vehicle* into the language of the *tenor*. For simplicity we will consider that the mapping function is defined for all elements of the *vehicle* language. Although this is usually not the case, we could, without loss of generality, extend the language of the *tenor* with those unmapped elements from the language of the *vehicle*, and extend the mapping function accordingly.

The two theories will be represented by generalized logic programs. The mapping function between the two languages will allow the construction of a function mapping theories of one language into theories of the other language. Formally we have:

Definition 5 (Metaphorical Program Mapping) *Let $\mathcal{K}_1$ and $\mathcal{K}_2$ be two arbitrary set of propositional variables whose names do not begin with a " not". Let $\psi_{1,2}$:$\mathcal{K}_1 \rightarrow \mathcal{K}_2$ be a function, mapping elements from $\mathcal{K}_1$ into elements of $\mathcal{K}_2$. Let $\mathcal{L}_1$ (resp. $\mathcal{L}_2$) be the language obtained from $\mathcal{K}_1$ (resp. $\mathcal{K}_2$). Let $\mathcal{P}_1$ (resp. $\mathcal{P}_2$) be the set of generalized logic programs in the language $\mathcal{L}_1$ (resp. $\mathcal{L}_2$). We define the metaphorical program mapping as the function $\Psi_{1,2}$:$\mathcal{P}_1 \rightarrow \mathcal{P}_2$ such that for every $P_1 \in \mathcal{P}_1$, $\Psi_{1,2}(P_1)$ is obtained by replacing every objective atom A (resp. default atom not A) appearing in a rule of P_1 by $\psi_{1,2}(A)$ (resp. not $\psi_{1,2}(A)$).* $\diamond$

Example 2 *Let $\mathcal{K}_1$ be:*

$$\{big, object, background\}$$

and $\mathcal{K}_2$ be:

$$\left\{ \begin{array}{c} long, motif, accompaniment, \\ isometric\_motet\_element \end{array} \right\}$$

corresponding to the example from the introduction. Let P_1 be[2]*:*

$$not\ big(X) \leftarrow object(X), background(X).$$

and P_2 be

$$long(X) \leftarrow motif(X), accompaniment(X), \\ isometric\_motet\_element(X).$$

Let $\psi_{1,2}$:$\mathcal{K}_1 \rightarrow \mathcal{K}_2$ be defined by:

$$\psi_{1,2}(big) = long$$
$$\psi_{1,2}(object) = motif$$
$$\psi_{1,2}(background) = accompaniment$$

If we apply $\Psi_{1,2}$ to P_1 we obtain $\Psi_{1,2}(P_1)$:

$$not\ long(X) \leftarrow motif(X), accompaniment(X). \diamond$$

Now that we have a process to transform a theory in one language (*vehicle*) into a corresponding theory in the language of the *tenor* (possibly extended with new elements), we need a way to combine this transformed theory with the theory representing the *tenor* to obtain a final theory.

This final theory will consist of the *tenor* theory together with those rules from the transformed *vehicle* theory that are not contradicted by the rules from the *tenor*. This can be seen as a process of accommodation where the rules from the transformed *vehicle* are combined with those from the *tenor*, provided they do not generate contradictions. This process is essentially equivalent to the inertia exerted on the rules of the initial program during an update. With this in mind, the definition of Metaphorical Program Update is:

Definition 6 (Metaphorical Program Update) *Let $\mathcal{K}_1$ and $\mathcal{K}_2$ be two arbitrary set of propositional variables whose names do not begin with a " not". Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be the languages obtained from $\mathcal{K}_1$ and $\mathcal{K}_2$ respectively. Let P_1 and P_2 be two generalized logic programs in the languages $\mathcal{L}_1$ and $\mathcal{L}_2$ respectively. Let $\psi_{1,2}$ be a function mapping elements from $\mathcal{K}_1$ into elements of $\mathcal{K}_2$. The metaphorical program update of P_1 by P_2 given $\psi_{1,2}$, denoted by $P_1 \odot P_2$ is given by $\Psi_{1,2}(P_1) \oplus P_2$.* $\diamond$

Example 3 *With P_1, P_2 and $\psi_{1,2}$ as in the previous example, the program $P_1 \odot P_2$ is:*

$$long(X)^{-}_{P_1} \leftarrow motif(X), accompaniment(X).$$
$$long(X)_{P_2} \leftarrow motif(X), accompaniment(X), \\ isometric\_motet\_element(X).$$

[2]Where, as usual, rules with variables stand for the set of their ground instantiations.

$$A^- \leftarrow A^-_{P_1}, not\ A_{P_2} \qquad A^- \leftarrow A^-_{P_2}$$
$$A^- \leftarrow not\ A_{P_1}, not\ A_{P_2} \qquad A \leftarrow A_{P_2}$$
$$A \leftarrow A_{P_1}, not\ A^-_{P_2} \qquad not\ A \leftarrow A^-$$

where A is a proposition from $\mathcal{K}_2$ and rules for A stand for their ground instances. Note that if we were to simply join the two programs, i.e. $\Psi_{1,2}(P_1) \cup P_2$, the two rules would produce a contradiction for X : $isometric\_motet\_element(X)$. If we, on the other hand, perform a metaphorical program update of P_1 by P_2, this contradiction no longer exists. From $P_1 \odot P_2$, we are able to conclude $long(X)$ for

$$\left\{ \begin{array}{c} X : isometric\_motet\_element(X), motif(X), \\ accompaniment(X) \end{array} \right\}$$

and not $long(X)$, otherwise, as intended. ◇

We go on by describing the semantics of the metaphorical program update $P_1 \odot P_2$ by providing a complete characterization of its stable models.. This will be done by means of a fixed-point equation defining the set of rules from the transformed *vehicle* that are rejected by the *tenor*, yielding the set of rules from the transformed *vehicle* that carry over to the final theory due to inertia.

Proposition 2 (Stable models of $P_1 \odot P_2$) *An interpretation N of the language $\widehat{\mathcal{L}_2}$ is a stable model of the metaphorical program update $P_1 \odot P_2$ if and only if N is the extension $N = \widehat{M}$ of a model M of P_2 that satisfies the condition:*

$$M = Least(\Psi_{1,2}(P_1) \cup P_2 - Rejected[M] \cup \\ \cup\ Defaults[M])$$

where $Rejected[M]$ and $Defaults[M]$ are as in Def.4, replacing P with $\Psi_{1,2}(P_1)$. ◇

The set $Defaults[M]$ contains default negations *not* A of all *unsupported* atoms A, i.e., atoms that have the property that the body of every clause from $\Psi_{1,2}(P_1) \cup P_2$ with the head A is false in M. Consequently, negation *not* A of these unsupported atoms A can be assumed by default. The set $Rejected[M] \subseteq \Psi_{1,2}(P_1)$ represents the set of transformed clauses of the *vehicle* program P_1 that are *rejected* (or contradicted) by the *tenor* program P_2 and the interpretation M.

5 Examples and Properties

In this section we will present a more elaborate example, based on the running example, and discuss some important characteristics of metaphorical program updates. We end the section with some considerations about the possible sources of contradiction within the presented framework.

Example 4 *Consider the following generalized logic program, representing some knowledge about the domain of Visual Arts*[3]*, P_1:*

$$not\ big(X) \leftarrow object(X), background(X). \quad (r_1)$$
$$tension(X) \leftarrow unbalanced(X). \quad (r_2)$$
$$contrast(X,Y) \leftarrow colour(X), colour(Y), \\ high\_value\_difference(X,Y). \quad (r_3)$$

Now consider another generalized logic program, representing some knowledge about the domain of Music, P_2:

$$long(X) \leftarrow motif(X), accompaniment(X), \\ isometric\_motet\_element(X). \quad (r_4)$$
$$tension(X) \leftarrow dissonant(X). \quad (r_5)$$

Let the metaphorical mapping be defined by the function $\psi_{1,2}$:$\mathcal{K}_1 \rightarrow \mathcal{K}_2$ such that:

$$\psi_{1,2}(big) = long$$
$$\psi_{1,2}(object) = motif$$
$$\psi_{1,2}(background) = accompaniment$$
$$\psi_{1,2}(tension) = tension$$
$$\psi_{1,2}(unbalanced) = dissonant$$
$$\psi_{1,2}(colour) = note$$
$$\psi_{1,2}(high\_value\_difference) = large\_interval$$
$$\psi_{1,2}(contrast) = contrast$$

If we apply $\Psi_{1,2}$ to P_1 we obtain $\Psi_{1,2}(P_1)$:

$$not\ long(X) \leftarrow motif(X), accompaniment(X). \quad (r_6)$$
$$tension(X) \leftarrow dissonant(X). \quad (r_7)$$
$$contrast(X,Y) \leftarrow note(X), note(Y), \\ large\_interval(X,Y). \quad (r_8)$$

Looking at the rules from $\Psi_{1,2}(P_1)$ and those from P_2, we can intuitively distinguish several paradigmatic cases:

- *rule r_6 will be a valid rule for those instances that are not covered by rule r_4 as explained during the running example;*
- *rule r_7 will not add anything to the metaphorical update for it is the same as rule r_5. This represents those cases where the translated rules from the* vehicle *are already present in the* tenor*;*
- *rule r_8 will bring not only new relations but also new concepts to the* tenor*. This represents the most interesting case with respect to creative reasoning.*

[3]The languages in which the programs are written are left implicit.

The program $P_1 \odot P_2$ is:

$$long(X)^{-}_{P_1} \leftarrow motive(X), accompaniment(X). \quad (11)$$

$$tension(X)_{P_1} \leftarrow dissonant(X). \quad (12)$$

$$contrast(X,Y)_{P_1} \leftarrow note(X), note(Y), large\_interval(X,Y). \quad (13)$$

$$long(X)_{P_2} \leftarrow motif(X), accompaniment(X), isometric\_motet\_element(X). \quad (14)$$

$$tension(X)_{P_2} \leftarrow dissonant(X). \quad (15)$$

*plus the corresponding **UR**, **IR** and **DR**. Note that rules (11) through (15), alone, are meaningless because they only have auxiliary literals ($L^{-}_{P_1}, L_{P_2}, ...$) as their conclusions and there are no rules for the literals in their premisses. It is through **UR**, **IR** and **DR** that we are able to determine the semantics of $P_1 \odot P_2$ with respect to the relevant literals. In the semantics of $P_1 \odot P_2$ we have:*

$long(X)$ for

$$\left\{ \begin{array}{c} X : isometric\_motet\_element(X), \\ motif(X), accompaniment(X) \end{array} \right\}$$

not $long(X)$ for

$$\left\{ \begin{array}{c} X : not\, isometric\_motet\_element(X), \\ motif(X), accompaniment(X) \end{array} \right\}$$

$contrast(X,Y)$ for

$$\left\{ \begin{array}{c} X,Y : note(X), note(Y), \\ large\_interval(X,Y) \end{array} \right\}$$

$tension(X)$ for

$$\{X : dissonant(X)\} \diamond$$

We believe it is interesting to draw the reader's attention to some properties emerging from the definition of metaphorical program updates, and their relation to known metaphor theory characteristics. The first one is related to the very basic intuition whereby metaphors bring new knowledge into the target domain. In fact, it is easy to see that in general we have that[4]:

$$SM(P_1 \odot P_2) \neq SM(P_2) \quad (16)$$

$$SM(P_1) \neq SM(P_2 \odot P_1) \quad (17)$$

The second and very important characteristic is that of directionality which, as explained before, means that each domain has a different role and its interchange, although possibly yielding an equally valuable metaphor, will not lead to the same meaning. If we have a bijective mapping function $\psi_{1,2}$ such that $\psi_{2,1} = \psi^{-1}_{1,2}$, then, in general, we have that

$$SM(P_1 \odot P_2) \neq SM(\Psi_{1,2}(P_2 \odot P_1)) \quad (18)$$

If we consider, for example, P_1 to represent a set of rules from the domain of Painting, and P_2 to represent a set of rules from the domain of Music, we could have an informal interpretation of the above properties reading as:

- a painter that becomes a musician would compose music different from that of a musician (16);
- a painter would paint differently from a musician that became a painter (17);
- a painter that becomes a musician would compose music different from that of a musician that becomes a painter (if he was to map his painting rules to music composition rules) (18).

It would be easy to check all these properties in the previous example.

In what contradiction is concerned, it is important to mention that the metaphorical program update ($P_1 \odot P_2$) only detects and deals with inconsistencies arising from rules from different domains. Other sources of inconsistencies can exist: P_1 can be contradictory, and so can be P_2; even in cases where P_1 and P_2 are not contradictory, $P_1 \odot P_2$ can be so, this happening when the contradiction is *'latent'* in one of the domains, and is *'brought alive'* by the metaphorical update, such as in the following example:

Example 5 *Consider the following program P_1:*

$$not\, hot(X) \leftarrow blue(X)$$
$$hot(X) \leftarrow red(X)$$

the metaphorical mapping $\psi_{1,2}$:

$$\psi_{1,2}(blue) = blues$$
$$\psi_{1,2}(red) = jazz$$
$$\psi_{1,2}(hot) = hot$$

and the program P_2

$$jazz(Miles) \leftarrow \qquad blues(Miles) \leftarrow$$

the metaphorical program update $P_1 \odot P_2$ is contradictory because both $hot(Miles)$ and $not\, hot(Miles)$ are derivable. $\diamond$

Nevertheless, be they important or not for our purposes, all contradictions can be detected by inspecting the truth value of the literals A^{-}, A_P, A^{-}_P, A_U, A^{-}_U, etc. of the program $P_1 \odot P_2$, and dealt with either by the semantics of $P_1 \odot P_2$ itself or by other known contradiction removal techniques, e.g. Alferes et al. (1995).

[4] Where by $SM(P)$ we mean the set of stable models of the program P, restricted to the relevant language ($\mathcal{L}_1$ or $\mathcal{L}_2$ depending on the case).

6 Conclusions and Future Work

Being *Metaphor* a common device for communication that uses interrelationships between different domains to assess new enriched mixed concepts, it is, as we believe, a powerful source for modelling *Creativity*. The ability to search for solutions in distant domains, apparently unrelated to the actual problem, is determinant for our creative abilities (Guilford, 1967; De Bono, 1970). Metaphor theories, such as Veale and Keane (1993), can be used as cross-domain bridge establishment methods, fundamental for knowledge integration within different domains.

In this paper we have explored the application of *Dynamic Logic Programming* to the problem of knowledge integration in metaphorical reasoning. The problem of resolving inconsistencies that may arise when knowledge from two different domains is combined, given a metaphorical mapping, is crucial, be it at the stage where we want to evaluate the appropriateness of the mapping function, or at a subsequent stage when we want to reason with the combined knowledge.

Quite interestingly, this combined knowledge becomes a new third domain which is not a crude sum of the original ones, but a blend of concepts and relationships among them which, in some cases, can yield potentially creative outcomes, much in the line of Turner and Fauconnier (1995).

We have proposed, in a formal and rigorous manner, a transformation that, by employing the principle of inertia on the rules of the *vehicle*, solves the problem of inter-domain inconsistencies. We have also characterized the models of the combined theory, all of this by means of the notions of *Dynamic Logic Programming*. Since DLP has also been implemented as a meta-interpreter (DLP System, 1998) running under XSB System (1999), this allows for not only theoretical but also practical reasoning.

This work is part of an ongoing larger project, *Dr. Divago*, whose final goal is to develop a system to perform automatic creative metaphorical reasoning.

In what future work is concerned, besides the integration of this framework within *Dr. Divago*, we are exploring some changes in the inertia rules to allow more flexibility and expressiveness, namely by permitting predominance of the *vehicle* over the *tenor*, among other possibilities. We are also exploring the development of this framework to enable for several distinct simultaneous metaphors, possibly with some preference relations amongst themselves. The mappings associated with these metaphors could be represented by generalized logic programs and the preferences could be refined by a combination of *Updates* with *Preferences* such as in Alferes and Pereira (2000).

Acknowledgements

We would like to thank Miguel Ferrand for his suggestions. The work of J. A. Leite was partially supported by PRAXIS XXI scholarship no. BD/13514/97. The work of J. A. Leite and L. M. Pereira was partially supported by PRAXIS XXI project MENTAL.

References

J. J. Alferes and C. V. Damásio and L. M. Pereira, *A Logic Programming System for Non-monotonic Reasoning*, Journal of Automated Reasoning (14):93-147, 1995

J. J. Alferes, J. A. Leite, L. M. Pereira, H. Przymusinski and T. C. Przymusinski. *Dynamic Logic Programming*. In A. Cohn, L. Schubert and S. Shapiro (eds.), Procs. of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98), Trento, Italy, pages 98-109. Morgan Kaufmann, June 1998.

J. J. Alferes, J. A. Leite, L. M. Pereira, H. Przymusinski and T. C. Przymusinski. *Dynamic Updates of Non-Monotonic Knowledge Bases*. To appear in The Journal of Logic Programming, 2000.

J. J. Alferes and L. M. Pereira. *Updates plus Preferences*. Technical Report, Dept. de Informática, New University of Lisbon, Portugal, 2000

Barnden, J. A. *An AI system for metaphorical reasoning about mental states in discourse*. In J-P. Koenig (Ed.), Conceptual Structure, Discourse and Language II, Stanford, Ca. 1997.

M. Black. *Models and Metaphor: studies in language and philosophy*. Ithaca, NY: Cornell University Press, 1962.

Edward De Bono. *Lateral Thinking: Creativity Step by Step*. Harper & Row, New York. 1970.

The DLP System, available from `centria.di.fct.unl.pt/~jja/updates`.

Umberto Eco. *Semiotics and the Philosophy of Language*. London: Macmillan Press. 1984.

Turner, M. and G. Fauconnier. (1995). *Conceptual Integration and Formal Expression*, Journal of Metaphor and Symbolic Activity 10(3).

Gentner, D., Falkenheiner, B. and Skorstad, J. Metaphor: *The Good, The Bad and The Ugly*. In Y. Wiks(Ed.), Theoretical Issues in Natural Language Processing. Hillsdale, NJ: Lawrence Erlbaum Associates. 1989.

J. P. Guilford. *The Nature of Human Intelligence*. New York: McGraw-Hill.1967.

M. Gelfond and V. Lifschitz. *The stable model semantics for logic programming*. In R. Kowalski and K. A. Bowen. editors. 5th International Logic Programming Conference, pages 1070-1080. MIT Press, 1988.

M. Gelfond and V. Lifschitz. *Classsical negation in logic programs and disjunctive databases*. New Generation Computing, 9:365-385, 1991.

B. Indurkhya. *Metaphor and Cognition*, Kluwer Academic Publishers, Dordrecht, 1992.

G. Lakoff and M. Johnson. *Metaphors We Live By*. Chigaco, Illinois: University of Chicago Press. 1980.

João A. Leite. *Logic Program Updates*. M.Sc. Dissertation, Universidade Nova de Lisboa, Portugal, 1997.

J. A. Leite and L. M. Pereira. *Generalizing updates: from models to programs*. In J.Dix, L.M. Pereira and T.C.Przymusinski (eds), Selected extended papers from the LPKR'97: ILPS'97 workshop on Logic Programming and Knowledge Representation, pages 224-246, Springer-Verlag, LNAI 1471, 1998

J. A. Leite and L. M. Pereira. *Iterated Logic Program Updates*. In J. Jaffar (ed.), Procs. of the 1998 Joint International Conference and Symposium on Logic Programming, Manchester, England, pages 265-278. MIT Press, June 1998.

Martin, J.H. *A computational model of metaphor interpretation*. Academic Press, 1990.

Francisco C. Pereira, *Modelling Divergent Production: a multi domain approach*. In Procs. of the Thirteenth European Conference of Artificial Intelligence, ECAI'98, Brighton, UK, 1998.

Francisco C. Pereira, *Construção Interactiva de Mapas Conceptuais*, M.Sc. Dissertation, University of Coimbra, Portugal, 2000

Richards, I. A. (1936). *The Philosophy of Rhetoric*. NY: Oxford University Press, 1936.

Tony Veale (1995). *Metaphor, Memory and Meaning: Symbolic and Connectionist Issues in Metaphor Comprehension*. PhD thesis submitted to Trinity College Dublin, 1995

Tony Veale. *'Just in Time' Analogical Mapping, An Iterative-Deepening Approach to Structure-Mapping* In Procs. of the Thirteenth European Conference of Artificial Intelligence, ECAI'98, Brighton, UK, 1998.

T. Veale and M. Keane. *A Connectionist Model of Semantic Memory for Metaphor Interpretation*, presented at the 1993 Workshop on Neural Architectures and Distributed AI.1993.

P. H. Winston. *Learning and Reasoning by Analogy*. Communications of the Association for Computing Machinery, 23(12), 1980.

The XSB Group. *The XSB logic programming system, version 2.0*, 1999. Available from `http://www.cs.sunysb.edu/~sbprolog`.

III. Techniques for Modelling Musical Creativity

Including Interval Encoding into Edit Distance Based Music Comparison and Retrieval

Kjell Lemström; Esko Ukkonen
Department of Computer Science; P.O.Box 26, FIN-00014 University of Helsinki, Finland
{klemstro,ukkonen}@cs.helsinki.fi

Abstract

A general framework for defining distance functions for monophonic music sequences is presented. The distance functions given by the framework have a similar structure, based on local transformations, as the well-known edit distance (Levenshtein distance) and can be evaluated using dynamic programming. The costs of the local transformations are allowed to be context-sensitive, a natural property when dealing with music. In order to understand transposition invariance in music comparison, the effect of interval encoding on some distance functions is analyzed. Then transposition invariant versions of the edit distance and the Hamming distance are constructed directly, without an explicit conversion of the sequences into interval encoding. A transposition invariant generalization of the Longest Common Subsequence measure is introduced and an efficient evaluation algorithm is developed. Finally, the necessary modifications of the distance functions for music information retrieval are sketched.

1 Introduction

The *translation-invariant* representation of integer sequences is useful in some application domains of string matching algorithms. In music retrieval, for example, it is often natural to require that the retrieval results are invariant with respect to pitch level transposition. In geometric shape detection the contour of a geometric object should be represented as a sequence that is invariant under rigid motions of the object.

Such an invariance can be achieved by converting the original 'absolute' sequence into relatively encoded form where each element of the sequence is encoded relative to its predecessor. The differences between successive elements are perhaps the simplest form of such an encoding. As an example, the absolute sequence $3, 7, 5, 5, 8, 7, 7, 5, 3$ is relatively encoded as $4, -2, 0, 3, -1, 0, -2, -2$.

In general string matching, the *edit distance* with its many variations is the most common measure to expose the (dis)similarity between two strings. The edit distance can be calculated by using dynamic programming; see e.g. (Crochemore and Rytter, 1994; Gusfield, 1997). The concept has also been adapted to music retrieval with relative encoding; see e.g. (Mongeau and Sankoff, 1990; Crawford et al., 1998).

The paper presents, in Section 3, a general framework for defining distance functions for monophonic music comparison and retrieval. In our framework one can use context-sensitive costs for the local transformations, that is a useful novel property when dealing with music. Analyzing the context of music usually requires a preprocessing phase which executes intelligent context analyzing programs. The result of such a preprocessing phase can be stored into some internal representation; see e.g. (Lemström and Laine, 1998).

In Section 4, we study the effect of the relative encoding on the values of some well-known and novel sequence distance functions. It turns out that typically there is no strong relation between distances measured using either absolute or relatively encoded sequences.

In Section 5, we find out, that an explicit relative encoding is not necessary. Transposition invariant distance functions can be constructed directly, by adopting transposition invariant cost functions for the local transformations used in the distance function. The cost functions now become context sensitive.

Section 6 introduces a new distance function for comparing music. It is a generalization of the classical longest common subsequence measure for comparing strings of symbols. One could call it as a 'Longest Common Hidden Melody' measure. An efficient evaluation algorithm is developed for this distance measure, improving the $O(m^2n^2)$ running time of the straightforward solution into $O(mn)$.

Finally in Section 7, we comment on using our distance functions in music information retrieval, that is, in finding the (approximate) occurrences of a short piece of music in a large music database.

2 Encodings of Music

We use a simplified representation of monophonic music that gives only the pitch levels of the notes, but not their durations. The pitch levels in semitones are given as inte-

ger numbers. Hence a piece of music containing n notes is represented as a sequence of n integers.

More formally, the integers that can be used in the sequences, form our *alphabet* Σ. In practice, we could for example have $\Sigma = \{0, 1, \ldots, 127\}$ (as in MIDI) or $\Sigma = \{0, 1, \ldots, 11\}$ (which reduces all notes into one octave). Any sequence $A = (a_1, a_2, \ldots, a_m)$ where each a_i is in Σ represents a piece of monophonic music. For simplicity, we will write such sequences in the sequel as $a_1 a_2 \cdots a_m$ instead of $(a_1, a_2, \ldots, a_m)$. The set of all sequences over Σ is denoted Σ^*. Hence $a_1 a_2 \cdots a_m$ is a member of Σ^*. The length of A is denoted $|A|$.

Two equally long sequences $A = a_1 \cdots a_m$ and $A' = a'_1 \cdots a'_m$ are *transpositions* of each other if there is an integer c such that $a'_1 = a_1 + c, a'_2 = a_2 + c, \ldots$, and $a'_m = a_m + c$. Then we write $A' = A + c$.

The *interval representation* of a sequence $A = a_1 \cdots a_m$ in Σ^* is a sequence

$$\overline{A} = (a_2 - a_1, a_3 - a_2, \ldots, a_m - a_{m-1}) = \overline{a}_1 \cdots \overline{a}_{m-1}.$$

Each symbol $\overline{a}_i$ in $\overline{A}$ is a difference between two integers in Σ. Hence $\overline{A}$ is a sequence in $\overline{\Sigma}^*$ where $\overline{\Sigma} = \{a - b \mid a, b \in \Sigma\}$ is the *interval alphabet.*

A *reduced interval alphabet* is often sufficient and useful in music information retrieval. For example, the so-called *octave equivalence* is achieved by using interval alphabet $\{a \bmod 12 \mid a \in \overline{\Sigma}\}$. If the intervals are known to be imprecise, like in query-by-humming systems, a rough classification of the intervals into possibly overlapping classes such as small, medium, and large upwards and downwards intervals might be sufficient (Ghias et al., 1995; Lemström and Laine, 1998).

The crucial property of the interval representation is that it is *transposition invariant.* This means, that if A and A' are transpositions of each other, then, obviously, they have the same interval representation, i.e.,

$$\overline{A} = \overline{A'}.$$

In music comparison and retrieval, it is often natural to have transposition invariant measures for the distance between sequences. Formally, a sequence distance function D is transposition invariant if

$$D(A, B) = D(A + a, B + b)$$

for any sequences A, B in Σ^* and any possible a, b in Σ.

Naturally, if the note durations are to be considered as well, an invariance under different tempi can be obtained by using a sequence $\hat{A}$, such that $\hat{A} = (\frac{a_2}{a_1}, \frac{a_3}{a_2}, \ldots, \frac{a_m}{a_{m-1}})$, that consists of the ratios of the original durations.

3 A Framework for Sequence Comparison

3.1 The General Scheme

We will introduce several different distance functions to measure the dissimilarity between sequences. All are variations of the well-known edit distance (also known as *Levenshtein distance* or *evolutionary distance*) widely used in approximate string matching. The underlying general framework for sequence distance functions is as follows; c.f. (Ukkonen, 1985).

To define a distance between sequences in Σ^*, one should first fix the set of *local transformations* $T \subseteq \Sigma^* \times \Sigma^*$ and a *cost function* w that gives for each transformation t a cost $w(t)$ which is a non-negative integer (or a real). Note that each t in T is a pair of sequences $t = (\alpha, \beta)$. It is convenient to extend w to all pairs (α, β) in $\Sigma^* \times \Sigma^*$: if $(\alpha, \beta) \notin T$, then we define $w(\alpha, \beta) = \infty$.

We usually write such a $t = (\alpha, \beta)$ as $(\alpha \rightarrow \beta)$ to emphasize another view on t: it can be seen as a rewriting rule that allows one to replace α by β inside a sequence that contains α.

The actual definition of the distance is based on the concept of *trace.* A trace gives a correspondence between two sequences. Formally, let $A = a_1 \cdots a_m$ and $B = b_1 \cdots b_n$ be sequences in Σ^*. A trace between A and B is formed by splitting A and B into equally many, say p, parts some of which may be empty sequences. Hence a trace can be written as:

$$\tau = (\alpha_1, \alpha_2, \ldots, \alpha_p; \beta_1, \beta_2, \ldots, \beta_p),$$

such that $A = \alpha_1 \alpha_2 \cdots \alpha_p$, and $B = \beta_1 \beta_2 \cdots \beta_p$, and each α_i, β_i is a possibly empty sequence over Σ. The trace suggests a transformation from A to B: part α_1 is transformed to β_1, α_2 to $\beta_2, \ldots$, and α_p to β_p. Stated otherwise, sequence B is obtained from A by local transformation steps $\alpha_1 \rightarrow \beta_1$, $\alpha_2 \rightarrow \beta_2, \ldots, \alpha_p \rightarrow \beta_p$.

Now the cost of the trace τ is $w(\tau) = w(\alpha_1 \rightarrow \beta_1) + \cdots + w(\alpha_p \rightarrow \beta_p)$. Note that if some $\alpha_i \rightarrow \beta_i$ is not in T, then $w(\tau)$ becomes ∞.

Finally, the distance between A and B, denoted $D_{T,w}(A, B)$, is defined as the minimum cost over all possible traces:

$$D_{T,w}(A, B) = \min\{w(\tau) \mid \tau \text{ is a trace between } A \text{ and } B\}.$$

3.2 Functions D_L and D_H

As an example, this scheme gives the familiar *unit-cost edit distance* (Levenshtein distance) if the local transformations are of the forms $a \rightarrow b$, $a \rightarrow \lambda$, and $\lambda \rightarrow a$ where a and b are any members of Σ, and λ denotes the empty sequence.[1] The costs are given as $w(a \rightarrow a) = 0$ for all a, and $w(a \rightarrow b) = 1$ for all $a \neq b$, and

[1] The local rules $a \rightarrow b, a \rightarrow \lambda$, and $\lambda \rightarrow a$ are often called 'replacement', 'deletion', and 'insertion'.

$w(a \to \lambda) = w(\lambda \to a) = 1$ for all a. We denote this edit distance function as D_L.

The *Hamming distance* is obtained exactly as D_L but the allowed local transformations are only $a \to b$ where a and b are any members of Σ, with cost $w(a \to a) = 0$ and $w(a \to b) = 1, a \neq b$. We denote the Hamming distance as D_H.

3.3 Distance Evaluation and Transformation Graph

Distances $D_{T,w}(A, B)$ are useful in practice, because they can efficiently be evaluated using dynamic programming. Such a procedure tabulates all distances $d_{ij} = D_{T,w}(a_1 \cdots a_i, b_1 \cdots b_j)$ between the prefixes of A and B. The distance table (d_{ij}), where $0 \leq i \leq m$ and $0 \leq j \leq n$, can be evaluated by proceeding row-by-row or column-by-column and using the recurrence

$$\begin{cases} d_{00} = 0 \\ d_{ij} = \min\{d_{r-1,s-1} + w(a_r \cdots a_i \to \\ \quad b_s \cdots b_j) \,|(a_r \cdots a_i \to b_s \cdots b_j) \in T\}. \end{cases} \quad (1)$$

Finally, d_{mn} equals the distance $D_{T,w}(A, B)$.

It is not difficult to see that the evaluation of $D_{T,w}(A, B)$ in this way may in the worst case take time at least proportional to m^2n^2 which is relatively slow[2]. In practice, fortunately, T is often very sparse which can be utilized to speed-up the dynamic programming. For example, the edit distance $D_L(A, B)$ can be evaluated in time proportional to mn (in time $O(mn)$, for short) from the well-known recurrence

$$\begin{aligned} d_{00} &= 0 \\ d_{ij} &= \min \begin{cases} d_{i-1,j} + 1 \\ d_{i,j-1} + 1 \\ d_{i-1,j-1} + (\text{if } a_i = b_j \text{ then 0 else 1}). \end{cases} \end{aligned}$$

An alternative view which sometimes is very useful, is given by interpreting (d_{ij}) as a weighted graph as follows. The d_{ij}'s form the nodes of the graph. Moreover, there is a weighted arc from any node $d_{r-1,s-1}$ to node d_{ij} if $(a_r \cdots a_i \to b_s \cdots b_j)$ belongs to T. This arc has weight $w(a_r \cdots a_i \to b_s \cdots b_j)$. Then $D_{T,w}(A, B)$ equals the smallest possible total weight of a path in this graph from d_{00} to d_{mn}; formal proof is a simple induction. We call such a path a *minimizing path*. The weighted graph is called the *transformation graph* and denoted $G_{T,w}(A, B)$.

3.4 Context-Sensitive Cost Functions

In our distance scheme as described above the cost $w(\alpha \to \beta)$ of each local transformation is independent of the context in which α and β occur in A and B. Transformation $\alpha \to \beta$ has always the same cost. When comparing musical sequences, however, a context-sensitive cost function seems sometimes useful. The cost of $\alpha \to \beta$ should depend on the (tonal, harmonic, rhythmic) context around α and β; see e.g. (Rolland and Ganascia, 1996; Coyle and Shmulevich, 1998).

This can be formalized by including the context in the definition of the cost function w. There are several possibilities as regards limiting the size of the context that can influence the cost. We can imagine that sometimes the total context of α and β, that is, the whole sequences A and B are relevant.

We suggest a parameterized limitation of the context. Assume that A can be written as $A = \mu\varphi\alpha\varphi'\mu'$ where $|\varphi| = u$ and $|\varphi'| = v$. Then α has (u, v) *context* (φ, φ'). Now we say that w is a *cost function with* (u, v) *context* if w is an integer valued function defined on $T \times \Sigma^u \times \Sigma^v \times \Sigma^u \times \Sigma^v$; here Σ^x denotes the set of sequences of length x over Σ. When using such a function w for evaluating the cost of a trace, the (u, v) context of each local rule is taken into account. If $\alpha \to \beta$ occurs in a trace such that the (u, v) context of α and β is (φ, φ') and (ϱ, ϱ'), respectively, then $w(\alpha \to \beta, \varphi, \varphi', \varrho, \varrho')$ is the cost associated with this particular occurrence of $\alpha \to \beta$.

The (u, v) context of each candidate transformation $\alpha \to \beta$ can easily be retrieved during dynamic programming when evaluating each d_{ij}. Hence using a cost function with (u, v) context is possible in the dynamic programming algorithm. The run time obviously gets slower but the overall architecture of the evaluation process remains the same.

It will turn out later in this paper that our transposition invariant sequence distance functions use cost functions with $(1, 0)$ context.

4 Absolute vs. Transposition Invariant Distance

The sequence distance functions given by our framework, such as D_L and D_H, can be applied as such both for comparing sequences in Σ^* ('absolute' sequences) and comparing their interval encoded versions in $\overline{\Sigma}^*$ ('relative' sequences). The functions induce two distances in this way. We say for sequences A, B in Σ^*, that $D_L(A, B)$ is the *absolute* edit distance and $D_L(\overline{A}, \overline{B})$ is the *transposition invariant* edit distance between A and B. When comparing music, the transposition invariant distance seems more natural except when we know *a priori* that A and B belong to the same key.

In this section we present some basic results, mostly on the relation between the absolute and the transposition invariant versions of distances D_L and D_H. However, let us start by introducing a *modulation function*. Let $A = a_1 \cdots a_m$. For any $l, 1 \leq l \leq m$, and any integer valued c, sequence

$$\mathcal{M}_l^c(A) = (a_1, \ldots, a_{l-1}, a_l + c, \ldots, a_m + c)$$

is called a *modulation* of A by c at l.

[2]Depending on the representation for T and w, the run time can be much slower.

Note that transposition is a special case: If B is a transposition of A, $B = \mathcal{M}_1^c(A)$.

The following theorem illustrates a strength of the interval encoding. Although one single modulation in the original sequences can change any number of notes, the (edit of Hamming) distance of the interval encoded versions stays small.

Theorem 1 *If B is a modulation of A then $D_L(\overline{A},\overline{B}) = D_H(\overline{A},\overline{B}) = 1$.*

Proof Let $B = \mathcal{M}_l^c(A)$. Hence $b_i = a_i$, for $1 \leq i < l$, and $b_i = a_i + c$ for $l \leq i \leq |A|$. Then obviously, $\overline{b}_i = \overline{a}_i$ for $1 \leq i < l-1$, $\overline{b}_{l-1} = \overline{a}_{l-1} + c$, and $\overline{b}_i = \overline{a}_i$ for $l \leq i < |A|$.

Sequences $\overline{A}$ and $\overline{B}$ differ in one position, hence the theorem follows. □

We continue with a technical lemma that points out the local configurations in the transformation graphs in which the transposition invariant distance can locally be larger than the corresponding absolute distance.

Distances D_H and D_L have similar properties, hence we analyze them together. Let D_E denote D_H or D_L. We need to compare the paths in $G_E(A,B)$ and in $G_E(\overline{A},\overline{B})$. Denote the nodes of $G_E(A,B)$ as $(d_{ij}), 0 \leq i \leq m, 0 \leq j \leq n$. Note that as $\overline{A}$ and $\overline{B}$ are one element shorter than A and B, the first row and column are missing in $G_E(\overline{A},\overline{B})$ as compared to $G_E(A,B)$. The subgraph of $G_E(A,B)$ with nodes $(d_{ij}), 1 \leq i \leq m, 1 \leq j \leq n$, has the same topological structure as $D_E(\overline{A},\overline{B})$ but some arc weights may differ. Let us denote this subgraph as $G'_E(A,B)$. Consider some directed path p in $G_E(A,B)$ from d_{00} to d_{mn}. Let p' be the restriction of p to $G'_E(A,B)$. Hence p' is a path from d_{rs} to d_{mn} for some r and s such that $r = 1$ or $s = 1$. Let e and f be two consecutive arcs of p'. We say that f starts a *0-block* of p', if e has weight 1 and f has weight 0.

Lemma 2 *Let g be an arc of p' such that g has weight 0 while the corresponding arc $\overline{g}$ in $G_E(\overline{A},\overline{B})$ has weight 1. Then g starts a 0-block in p'.*

Proof. If g does not start a 0-block then the arc g' just before g on p' must have weight 0, too. The only local transformations with weight 0 that are available for distance D_E are the identity transformations of the form $a \to a$. But this means that for some i and j, g is an arc from $d_{i-1,j-1}$ to d_{ij} and g' is an arc from $d_{i-2,j-2}$ to $d_{i-1,j-1}$, and $a_i = b_j$ and $a_{i-1} = b_{j-1}$. Hence $a_i - a_{i-1} = b_j - b_{j-1}$ which means that $\overline{g}$ has weight 0 in $G_E(\overline{A},\overline{B})$. Hence, if $\overline{g}$ has weight 1, g must start a 0-block in p'. □

Theorem 3 $D_E(\overline{A},\overline{B}) \leq 2 \cdot D_E(A,B)$.

Proof. Let p be the minimizing path in $G_E(A,B)$ from d_{00} to d_{mn}, and let p' be its restriction to $G'_E(A,B)$. Path p' can contain at most $D_E(A,B)$ 0-blocks because there must be before each block an arc with weight 1. Then by Lemma 2, path $\overline{p'}$ that corresponds to p' in $G_E(\overline{A},\overline{B})$ can have at most $D_E(A,B)$ 1-arcs more than p'.

Moreover, let $\overline{p''}$ be the path in $G_E(\overline{A},\overline{B})$ from $\overline{d}_{11}$ to the start node of $\overline{p'}$. A simple case analysis shows that the total weight of $\overline{p''}$ is at most the weight of the subpath of p that leads from d_{00} to the start node of p'. In summary, this means that the weight W of path $\overline{p''}\,\overline{p'}$ in $G_E(\overline{A},\overline{B})$ from $\overline{d}_{11}$ to $\overline{d}_{mn}$ is $\leq 2 \cdot D_E(A,B)$. The theorem follows as $\overline{p''}\,\overline{p'}$ is not necessarily the minimizing path in $G_E(\overline{A},\overline{B})$ and hence $D_E(\overline{A},\overline{B}) \leq W$. □

The following theorem gives bounds for the difference $D(A,B) - D(\overline{A},\overline{B})$ when D is D_L or D_H. We also consider a distance $D_{H'}$ which is the Hamming distance augmented with a novel local transformation $ab \to cd$ (called *compensation*), where a, b, c, and d are members of Σ such that $a+b = c+d$. Moreover, $w(ab \to cd) = 1$.

To understand the intuition behind the compensation rule, consider the effect of a single replacement. Let A and B differ only by one replacement (also called a *mismatch*) which we denote $B = \mathcal{V}_l(A)$ where l refers to the mismatching position. Then $D_H(A,B) = 1$ while $D_H(\overline{A},\overline{B}) = 2$ because a mismatch changes two intervals. By adding the compensation rule this antisymmetry is at least partially relieved, because $D_{H'}(\overline{A},\overline{B}) = 1$.

Theorem 4 *a) Let $|A| = |B| = n$.*
Then $-\lfloor \frac{n-1}{2} \rfloor \leq D_H(A,B) - D_H(\overline{A},\overline{B}) \leq n$.

b) Let $|A| = |B| = n$. Then
$-\lfloor \frac{n-1}{3} \rfloor \leq D_H(A,B) - D_{H'}(\overline{A},\overline{B}) \leq n$.

c) Let $m = |A| \leq |B| = n$. Then
$1 - m \leq D_L(A,B) - D_L(\overline{A},\overline{B}) \leq n$.

Proof. To prove the upper bounds in all the cases, we note that, clearly, the bound cannot be larger than n because the distances between A and B as well as between $\overline{A}$ and $\overline{B}$ always belong to the range $0, \ldots, n$. To show that the bound is tight, let $A = aa\cdots a$ and $B = bb\cdots b$ for some $a \neq b$ and $|A| = |B| = n$. Then $D_H(A,B) = D_L(A,B) = n$ while $D_H(\overline{A},\overline{B}) = D_{H'}(\overline{A},\overline{B}) = D_L(\overline{A},\overline{B}) = 0$. Hence the bound cannot be smaller than n.

The lower bound follows from the fact that the number of 0-blocks of Lemma 2 is at most $\lfloor \frac{n-1}{2} \rfloor$ in the case of Hamming distance D_H; at most $\lfloor \frac{n-1}{3} \rfloor$ in the case of modified Hamming distance $D_{H'}$; and at most $m-1$ in the case of Levenshtein distance D_L.

All the bounds are again tight. For D_H consider sequences $A = (0,1,0,1,\ldots)$ and $B = (0,2,0,2,\ldots)$; for $D_{H'}$ sequences $A = (0,0,1,0,0,1,\ldots,0,0,1)$ and $B = (0,0,2,0,0,2,\ldots,0,0,2)$; and for D_L sequences

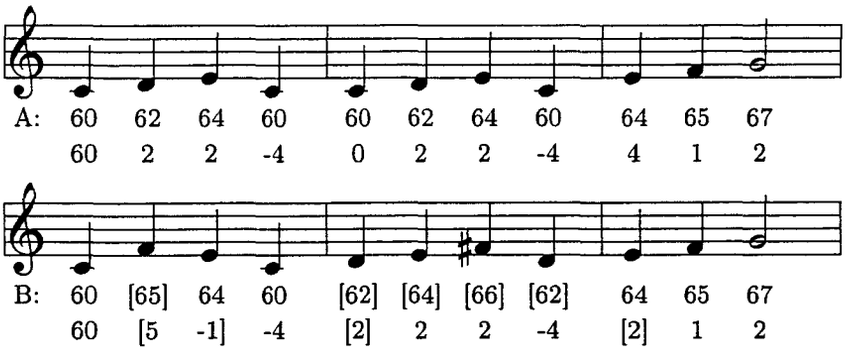


Figure 1: $D_H(A,B)$ vs. $D_{H'}(\overline{A},\overline{B})$ in music comparison. The sequences below A and B correspond to $\overline{A}$ and $\overline{B}$, respectively.

$A = (0,0,\ldots,0)$ and $B = (1,0,1,0,\ldots,1,0)$ such that $|B| = 2 \cdot |A|$. □

Example 1 *In Fig. 1, the absolutely encoded A and B are comprised of MIDI values. The two sequences are almost identical: there is only one mismatch and two modulations ($B = \mathcal{M}_9^{-2}(\mathcal{M}_5^2(\mathcal{V}_2(A)))$). We have enclosed in brackets the locations where an editing operation is needed. In $D_{H'}(\overline{A},\overline{B})$, the mismatch is detected by a compensation ($\overline{a}_2 + \overline{a}_3 = \overline{b}_2 + \overline{b}_3 = 4$) and the modulations by a replacement, thus $D_{H'}(\overline{A},\overline{B}) = 3$. However, $D(A,B) = 5$ because the mismatch is detected by a replacement, but the first modulation has shifted all the values until the other 'normalizing' modulation. Note, that after being lost 'B catches' immediately the correct tune at the second modulation ($a_9 = b_9$), while '$\overline{B}$ realigns' itself one step later.* □

5 Implicit Interval Encoding

We noticed in Section 4 that each distance function for absolute sequences also gives a transposition invariant distance. This is achieved simply by at first converting the sequences into interval encoding. We now complement this by observing that an explicit conversion is not necessary. The cost function for local transformations can be defined such that the resulting distance becomes transposition invariant.

Since the conversion is not needed anymore, some shortcomings of the relative encoding are avoided. When the intervals are calculated 'on-the-fly' from absolute sequences, a deletion or an insertion does not transpose the rest of a sequence as in the case when working with relative encoding.

We restrict the consideration to distances D_H and D_L only. The *transposition invariant unit-cost edit distance* $\overline{D}_L(A,B)$ uses the local transformations $a \to b$, $a \to \lambda$, and $\lambda \to a$ as the distance $D_L(A,B)$. The costs are given as $\overline{w}(a \to \lambda) = \overline{w}(\lambda \to a) = 1$. The cost for $a \to b$ will now be context-sensitive. Let a' be the symbol of A just before a and b' the symbol of B just before b', that is, the $(1,0)$ contexts of a and b are (a',λ) and (b',λ). Then define

$$\overline{w}(a \to b, a', \lambda, b', \lambda) = \begin{cases} 0, \text{if } a - a' = b - b' \text{ or} \\ \quad a' \text{ or } b' \text{ is missing,} \\ 1, \text{if } a - a' \neq b - b'. \end{cases}$$

The recurrence for evaluating the prefix distance table (d_{ij}) for $\overline{D}_L(A,B)$ becomes

$$\begin{aligned} d_{00} &= 0 \\ d_{ij} &= \min \begin{cases} d_{i-1,j} + 1 \\ d_{i,j-1} + 1 \\ d_{i-1,j-1} + (\text{if } a_i - a_{i-1} = b_j - b_{j-1} \\ \qquad \text{then 0 else 1}). \end{cases} \end{aligned}$$

Obviously, this table can be evaluated in time $O(mn)$.

Similarly, the *transposition invariant Hamming distance* $\overline{D}_H(A,B)$ uses local transformations $a \to b$ whose cost is defined exactly as for $\overline{D}_L(A,B)$ above.

By induction over the corresponding prefix distance tables (d_{ij}) one easily proves the following theorem which implies that $\overline{D}_L$ and $\overline{D}_H$ really are transposition invariant distance functions according to the definition given in Section 2.

Theorem 5

$$\begin{aligned} \overline{D}_L(A,B) &= D_L(\overline{A},\overline{B}) \\ \overline{D}_H(A,B) &= D_H(\overline{A},\overline{B}). \end{aligned}$$

This also includes that the prefix distance table (d_{ij}) for $\overline{D}_L$ is of the same general form as the table for the standard unit-cost edit distance. Hence the very fast bit-parallel algorithms designed for the edit distance can be used for evaluating the transposition invariant unit-cost edit distance as well.

Finally we remark that using both the absolute and relative costs of local transformations simultaneously is possible and seems to give useful distance functions. For example, in the standard unit-cost edit distance we could

give cost 0 to $a \rightarrow b$ if $a = b$, as usual, but also if $a \rightarrow b$ occurs in a context in which the interval for a equals the interval for b. The recurrence becomes

$$\begin{aligned} d_{00} &= 0 \\ d_{ij} &= \min \begin{cases} d_{i-1,j} + 1 \\ d_{i,j-1} + 1 \\ d_{i-1,j-1} + (\text{if } a_i = b_j \text{ or } a_i - a_{i-1} = \\ \qquad\qquad b_j - b_{j-1} \text{ then 0 else 1}). \end{cases} \end{aligned}$$

We denote the resulting distance function as D_N. As D_N has the same local transformations as D_L and $\overline{D}_L$, with possibly smaller costs, it follows that $D_N(A,B) \leq D_L(A,B)$ and $D_N(A,B) \leq \overline{D}_L(A,B)$ for all A and B. On the other hand, D_N is not transposition invariant.

6 Transposition Invariant LCS

The *longest common subsequence* $LCS(A,B)$ of two sequences A and B can be used for measuring the similarity of A and B: the longer is $LCS(A,B)$, the more similar are the sequences A and B.

To define $LCS(A,B)$, we say that sequence C is a *subsequence* of a sequence A if C can be obtained from A by deleting zero or more symbols. Then $LCS(A,B)$ is the longest sequence that is a subsequence of both A and B. For music comparison and retrieval we need a transposition invariant generalization of this concept.

We say that a sequence C in Σ^* is the *longest common transposition invariant subsequence* of two sequences A and B in Σ^* if C is the longest possible sequence such that $\overline{C} = \overline{A'}$ and $\overline{C} = \overline{B'}$ and A' is a subsequence of A and B' is a subsequence of B. Then we write $C = LCTS(A,B)$. Note that $LCTS(A,B)$ is unique only up to arbitrary transposition: if $C = LCTS(A,B)$ then any $C + c$ is $LCTS(A,B)$.

The sequence $C = LCTS(A,B)$ can be seen in musical terms as the longest common melody that is hidden in both A and B. To obtain C, we must delete the smallest number of elements from A and B such that the remaining two sequences are identical after some transposition, that is, their interval encoded representations are identical. This seems like a natural concept in the context of music comparison. We can think, for example, that the deleted elements are musical decorations used differently in the two variations A and B of the same melody C [3].

Let $D_{LCS}(A,B)$ denote the total number of deletions needed to obtain $LCS(A,B)$ from A and B. Then it is well-known that

$$|LCS(A,B)| = \frac{|A| + |B| - D_{LCS}(A,B)}{2},$$

where $D_{LCS}(A,B)$ is a distance function that is given by our general scheme by using local transformations $a \rightarrow \lambda$, $\lambda \rightarrow a$, and $a \rightarrow a$ with costs $w(a \rightarrow \lambda) = w(\lambda \rightarrow a) = 1$, and $w(a \rightarrow a) = 0$.

Similarly, it turns out that

$$|LCTS(A,B)| = \frac{|A| + |B| - D_{LCTS}(A,B)}{2},$$

where $D_{LCTS}(A,B)$ is a distance function again given by our scheme. Now any rule $\alpha \rightarrow \beta$ where α, β are *non-empty* sequences in Σ^* is a local transformation. The associated cost function w_{LCTS} has $(1,0)$ context. Let a be the last symbol of α and b the last symbol of β and let the $(1,0)$ context of α and β be (a', λ) and (b', λ), respectively.

Then we define

$$w_{LCTS}(\alpha \rightarrow \beta, a', \lambda, b', \lambda) = \begin{cases} k-1+l-1, \text{if } a-a' = b-b' \\ k+l, \text{if } a-a' \neq b-b', \end{cases}$$

where $k = |\alpha|$ and $l = |\beta|$.

The recurrence for tabulating (d_{ij}) to get $D_{LCTS}(A,B)$ is

$$\begin{aligned} d_{00} &= 0 \\ d_{ij} &= \min_{1 \leq k \leq i; 1 \leq l \leq j} \{d_{i-k,j-l} + \\ &\qquad w_{LCTS}(\alpha \rightarrow \beta, a_{i-k}, \lambda, b_{j-l}, \lambda)\}, \end{aligned}$$

where $\alpha = a_{i-k+1} \cdots a_i$ and $\beta = b_{j-l+1} \cdots b_j$. To evaluate d_{ij} directly from the recurrence needs $O(ij)$ operations, hence the total time requirement becomes as high as $O(m^2n^2)$. We show next how to reduce the computation of $D_{LCTS}(A,B)$ to a repeated computation of the distance D_{LCS}.

The key observation is, that $LCTS(A,B)$ must be equal to a $LCS(A, B+c)$ for some suitably selected constant c.

This gives the following solution: For all c in $\overline{\Sigma}$, compute $D_{LCS}(A, B+c)$ with the standard $O(mn)$ algorithm. Then $|D_{LCTS}(A,B)| = \max_c\{|D_{LCS}(A, B+c)|\}$. This method takes time proportional to $|\overline{\Sigma}| \cdot mn$ which is $O(mn)$ because $|\overline{\Sigma}|$ is independent of m and n. Hence we have:

Theorem 6 *$D_{LTCS}(A,B)$ and hence $|LCTS(A,B)|$ can be computed in time $O(mn)$.*

In Fig. 2 we give an example of calculating an LCTS between two musical sequences (the sequences are obtained from (Cambouropoulos et al., 1999); the lower sequence is transposed from the original key of C major to D major).

7 Music Retrieval

In the content-based information retrieval problem for monophonic music we are given a long database $S = s_1 s_2 \cdots s_n$ of monophonic music and a relatively short

[3] To really achieve this goal needs further elaboration of the LCTS model that goes beyond the present paper.

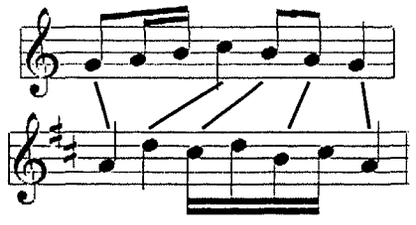


Figure 2: An example of finding $LCTS(A, B)$. The applied $(a \rightarrow a + c)$ rules are illustrated by lines. In this case $|LCTS(A, B)| = 5$, and $c = 2$.

'key' $P = p_1 \cdots p_m$. Formally, both S and P are sequences in Σ^*. The problem is to find the locations in S where P occurs as a subsequence. One might be interested in finding for example exact occurrences, transposed occurrences or maybe somehow approximate occurrences.

The framework of Section 3 is for comparing entire sequences. It is, however, easy to modify it using a well-known trick, originally presented by Sellers (1980), for our retrieval problem. Now P should basically be compared against *all subsequences* of S to find the best match. The dynamic programming algorithm almost as such will do this, only a slight change is needed in the initialization of the table (d_{ij}).

More precisely, let $D_{T,w}$ be a distance function given by the scheme of Section 3. Evaluate table $(d_{ij}), 0 \leq i \leq m, 0 \leq j \leq n$, as in the recurrence (1) but use initialization

$$d_{0j} = 0$$

for $0 \leq j \leq n$. Then the music retrieval results can be read from the last row $d_{mj}, 0 \leq j \leq n$, of the table (d_{ij}). Value d_{mj} gives the smallest possible value of $D_{T,w}(P, P')$ where P' is any subsequence (substring) of S that ends at location j. The sequence P' can be uncovered using (d_{ij}), too.

The length n of the database S can be very large. Therefore it is of crucial importance to find fast implementations for the evaluation of (d_{ij}). The fastest algorithms currently known are based on so-called bit-parallelism (Baeza-Yates and Gonnet, 1992). Bit-parallelism can give a speed-up up to by factor W where W is the length (in bits) of the computer word used. Unfortunately, such algorithms have quite limited applicability: strong restrictions on the set T of local transformations and on the cost function w, seem necessary.

The unit-cost edit distance D_L is an example of a distance function for which bit-parallel implementation is possible. The fastest such algorithms currently known are due to Myers (1998), and Navarro and Raffinot (1998). It turns out that Myers' algorithm can be modified (at least) to the distance function D_N and $\overline{D}_L$ defined in Section 5. We have implemented these in our prototype MIR system under development (Lemström and Perttu, 2000). The algorithms can reach a scanning speed exceeding 10^7 notes per second on current Pentium II computers.

8 Conclusion

We have considered the problem of measuring the distance between two sequences A and B in the context of monophonic music comparison and retrieval. The encoding that we have used is a simplified representation of monophonic music; only the pitch levels are present in the encoding.

We have presented a general framework for sequence comparison. The framework deals with variations of the well-known edit distance measure. Moreover, in our framework one can use context-sensitive cost functions, which we believe that is a very important property in this application area. We also introduced the concept of a transposition invariant distance function and presented some examples of such functions.

Currently we are working on various bit-parallel algorithms for music information retrieval, based on the distance functions presented in this paper.

References

R. Baeza-Yates and G. H. Gonnet. A new approach to text searching. *Communications of the ACM*, 35(10): 74–82, 1992.

E. Cambouropoulos, T. Crawford, and C. S. Iliopoulos. Pattern processing in melodic sequences: Challenges, caveats & prospects. In *Proceedings of the AISB'99 Symposium on Musical Creativity*, pages 42–47, 1999.

E. Coyle and I. Shmulevich. A system for machine recognition of music patterns. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 3597–3600, Seattle, WA, 1998.

T. Crawford, C. S. Iliopoulos, and R. Raman. String matching techniques for musical similarity and melodic recognition. *Computing in Musicology*, 11: 71–100, 1998.

M. Crochemore and W. Rytter. *Text Algorithms*. Oxford University Press, 1994.

A. Ghias, J. Logan, D. Chamberlin, and B. C. Smith. Query by humming - musical information retrieval in an audio database. In *ACM Multimedia 95 Proceedings*, pages 231–236, San Francisco, California, 1995.

D. Gusfield. *Algorithms on strings, trees, and sequences*. Cambridge University Press, 1997.

K. Lemström and P. Laine. Musical information retrieval using musical parameters. In *Proceedings of the 1998*

International Computer Music Conference, pages 341–348, Ann Arbor, Michigan, 1998.

K. Lemström and S. Perttu. SEMEX - an efficient music retrieval prototype. Manuscript (in preparation), 2000.

M. Mongeau and D. Sankoff. Comparison of musical sequences. *Computers and the Humanities*, 24:161–175, 1990.

G. Myers. A fast bit-vector algorithm for approximate string matching based on dynamic programming. In *Proceedings of Combinatorial Pattern Matching*, pages 1–13, Piscataway, New Jersey, 1998.

G. Navarro and M. Raffinot. A bit-parallel approach to suffix automata: Fast extended string matching. In *Proceedings of Combinatorial Pattern Matching*, pages 14–33, Piscataway, New Jersey, 1998.

P.-Y. Rolland and J.-G. Ganascia. Automated motive oriented analysis of musical corpuses: a jazz case study. In *Proceedings of the 1996 International Computer Music Conference*, pages 240–243, Hong Kong, 1996.

P. H. Sellers. The theory and computation of evolutionary distances: Pattern recognition. *Journal of Algorithms*, 1(4):359–373, 1980.

E. Ukkonen. Algorithms for approximate string matching. *Information and Control*, 64:100–118, 1985.

A prototype direct manipulation music tool based on Lerdahl's Pitch Spaces

Matt Smith
School of Computing Science
Middlesex University - Bounds Green
London N11 2NQ - United Kingdom
T: (+44) (0)181 362 6727
F: (+44) (0)181 362 6411
m.r.smith@mdx.ac.uk

Abstract

Lerdahl (1988) suggests a model of 'Pitch Spaces' that treats pitches, chords and regions in a single framework. In this paper we will both summarise Lerdahl's arguments for the relative strengths of the model over previous cognitive models of harmonic knowledge, and we will present an extension to the model facilitating the design and implementation of a computer-based tool for harmonic analysis and composition for musical novices. We describe the current prototype of our computer music tool, and present plans for further development and evaluation of the tool with musical novices.

1 Introduction

Lerdahl (1988) suggests a model of 'Pitch Spaces' that treats pitches, chords and regions in a single framework. In this paper we will both summarise Lerdahl's arguments for the relative strengths of the model over previous cognitive models of harmonic knowledge, and we will present an extension to the model facilitating the design and implementation of a computer-based tool for harmonic analysis and composition for musical novices. Lerdahl's model, and the work presented in this paper are based around the Western Tonal Music system of harmony.

1 1 Motivation for another pitch space framework

Lerdahl's pitch space framework is not based on a SYMMETRICAL topological model - unlike those such as Longuet-Higgins (1962) and Shepard (1982). Lerdahl's states that a weakness of topological models is an over emphasis on symmetry - therefore misrepresenting the non-symmetrical aspects of the diatonic system. The pitch space framework is constructed with to be able to model the same asymmetry as the chords and keys it is modelling. Another strength of Lerdahl's model is that it is able to model more than one level of pitch representation in a single framework - previous systems have modelled either pitch classes (such as Balzano 1982 or Shepard 1982), or tonal regions (for example Weber 1824 and Schoenberg 1911/1978). Although one existing tonal framework does model multiple levels of pitch description - Longuet-Higgins (1962) - that system is applied to tonal regions (derived from pitch classes) and cannot be used to describe pitch proximity.

Previous work cited by Lerdahl as influential on the development of Pitch Spaces providing descriptions of pitch classes, chord spaces and tonal regions included Krumhansl (1979 & 1983), Krumhansl, Bharucha & Kessler (1982) and Krumhansl & Kessler (1982).

2 Description of Lerdahl's pitch space framework

Suggested by the chromatic, diatonic and triadic 'overlearning' ideas of Deutsch & Feroe (1984) and Deutsch (1982). Lerdahl's framework is a hierarchy of five spaces. The hierarchy is such that each level is made up of a subset of those pitch classes from the level immediately below. The five spaces are shown in Table 1.

Table 1: Names of pitch spaces

Level	Name
a	Octave space
b	Open fifth space

c	Triadic space
d	Diatonic space
e	Chromatic space

Important points Lerdahl makes about the spaces are as follows:
q except for the chromatic space, the spaces describe the asymmetric patterns appropriate for diatonic music
q the diatonic space is DIRECTLY represented in the framework (unlike the symmetrical frameworks mentioned earlier)
q the pitch space framework allows unified treatment of pitch class, chord and regional proximity

Lerdahl uses the Roman-numeral notation of chord / region - where we shall represent the region numeral in parentheses. For example. I/(I) is the pitch space for the tonic chord (say C major), in the region of the tonic (the C major diatonic region).

The choice of the tonic as C is arbitrary, and could be any other pitch class. The numeric forms for the pitch space I/(I) is shown in Table 2.

Table 2: Pitch space for I/(I)

Space	**Pitch Class**											
a	0											
b	0							7				
c	0				4			7				
d	0		2		4	5		7		9		b
e	0	1	2	3	4	5	6	7	8	9	a	b
e.d.	**0**	**4**	**3**	**4**	**2**	**3**	**4**	**1**	**4**	**3**	**4**	**3**

The final row, ED, is the 'embedding distance' - this is a measure of how far from the octave space a given pitch class is for a given pitch space. This distance shifts for a given chord and region. The shallower the embedding (such as 0 and 7 in I/(I)) the more important the pitch class harmonically for a given space.

Lerdahl explains this vertical embedding distance measure in terms of 'skip' and 'step':
"In traditional usage a STEP occurs between adjacent members of the chromatic or diatonic scales (a chromatic or diatonic step), and an ARPEGGIATION takes place between adjacent members of a triad. It is more illuminating, however, to thing of an arpeggiation as stepwise motion in triadic space [space c]. A leap of two octaves, on the other hand, is a skip in octave space [space a]. In sum, a step is adjacent motion along any level of the hierarchy, and a skip is non-adjacent motion -- two or more steps -- along any level."
(Lerdahl, 1988, pp. 321-322)

Therefore, using Lerdahl's definition of step and skip, the proximity of two pitches in a given pitch space (e.g. I/(I)) can be measured as a 'step distance' by the number of steps left or right at a given level to get from one pitch to another - e.g. in I/(I) from p0 to p4 is one step in triadic space, two steps in diatonic space and four steps in chromatic space. Chord proximity can be calculated using two factors: the diatonic circle of fifths and the number of common tones between the two chords. Lerdahl describes how each of these factors can be modelled via his pitch spaces. He presents the 'chord circle rule', defined as instruction to "move the pcs [pitch classes] at levels a-c four diatonic steps to the right of left (mod 12) on level d" (p. 322). Thus there is no change to the diatonic or chromatic spaces when modelling the chord circle.

The circle of fifths
[pc0 (I) - pc7 (V) - pc2 (ii) - pc9 (vi) - pc4 (iii) - pcb)IV) - pc0 (I) and so on] appears as a sequence of pitch spaces when the chord circle rule is successively applied as shown from I/(V) to I/(ii) in tables 3 and 4.

Table 3: Pitch space for V/(I)

Space	Pitch Class											
c	0		2					7				b
d	0		2		4	5		7		9		b
e	0	1	2	3	4	5	6	7	8	9	a	b

Table 4: Pitch space for ii/(I)

Space	Pitch Class											
a			2									
b			2							9		
c			2			5				9		
d	0		2		4	5		7		9		b
e	0	1	2	3	4	5	6	7	8	9	a	b

Lerdahl goes on to define a common tone distance between any two chords based on the number of distinct PCs and the shortest number of steps on the circle of fifths. He goes further, and defines a measure of chord proximity across regions using a rule that gives a chromatic circle of fifths, and a measure of region proximity and how seventh and minor chords can be modelled in the framework.

The hierarchical and numerical nature of the pitch spaces, and the simplicity of the measurement of pitch, chord and region proximity suggest the use of this framework for computational modelling. The pitch space formalism has strong explanatory power, and as Lerdahl goes on to discuss, appears to correlate with experimental results investigating pitch class stability (see Krumhansl 1979, and Krumhansl & Shepard 1979), multi-dimensional scaling of diatonic triads (Krumhansl, Bharucha & Kessler, 1982) and abstract region spaces (Krumhansl & Kessler, 1982).

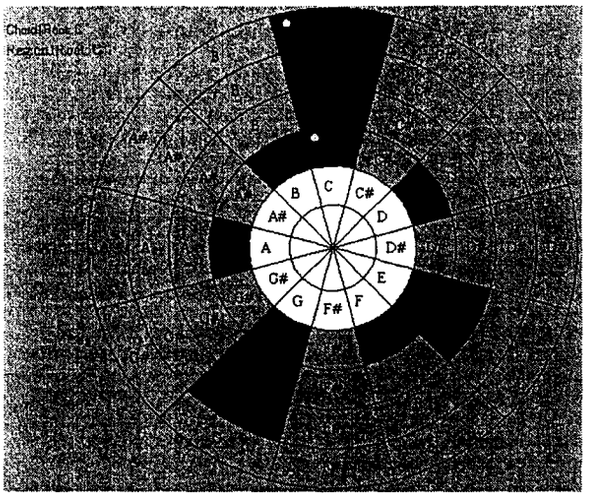


Figure 2: I/(**I**) as C/(C)

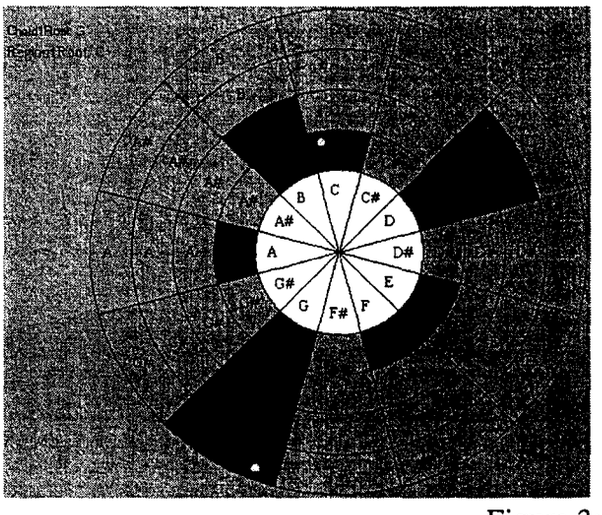


Figure 3: V/(**I**) as G/(C)

As you can see from the difference between Figure 1 and Figure 2 (and full versions in the web browser in Figures 9 and 10) the prototype allows the user to choose whether to see pitch class numbers (0..b or 0..11) or the note letters assuming pitch class 0 is C.

The user currently has simply facilities such as stepping (rotating) the major chord circles (levels a, b and c), and/or stepping the diatonic region circle (level d). The chromatic circle is left unchanged[1].

5 Proposed experimental evaluation

We are in the process of extending our prototype for use in experiments with novice musicians for the support of simple harmonic analysis and composition tasks. If Lerdahl's claims are correct, and the asymmetry of his model provides a good cognitive fit with human harmonic problem solving, we expect to find positive results from our trials, in comparisons with alternatives such as Holland's (1989) "Harmony Space" interactive computer tool.

In the next few sections we shall present some examples of the kinds of questions musical novices might be asked, and how they might be able to answer them using the tool.

1. Although it might make sense, and be more consistent, to allow the chromatic circle to be rotated to indicate when region **I** is associated with a note other than C.

5 1 Chords in a region

The musical novice might be presented with the system set up with Cmaj chord in the region Cmaj (as in Figure 2). They could be asked the following question:

In addition to Cmaj what are the other major chords likely to sound nice in this region?

A likely way to try to answer this question would be to keep the chromatic and diatonic circles (spaces 'd' and 'e') as they are, and to rotate the chord circles ('a', 'b' and 'c') to find chords which have common pitch classes with the current diatonic region (i.e. Fmaj and Gmaj).

Rotating clockwise from Cmaj, the student would first come to C#/(**C**) (see Figure 4). Clearly this chord does not share all notes with the region Cmaj, since neither C# nor G# as in Cmaj. Continuing to step the major chord shape (rotate the outer three circles) around the chromatic or diatonic circles the student would first come to chord Fmaj (see Figure 5) then Gmaj (see Figure 3) and find that they, in addition to Cmaj, are the only three major chords that have all notes common with the Cmaj region.

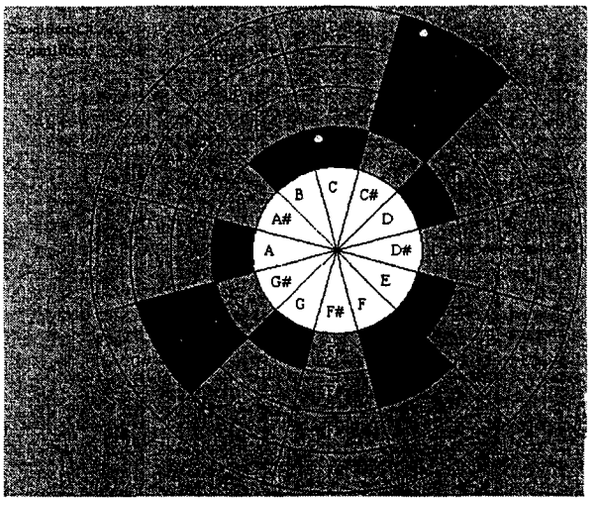


Figure 4: C#/(C)

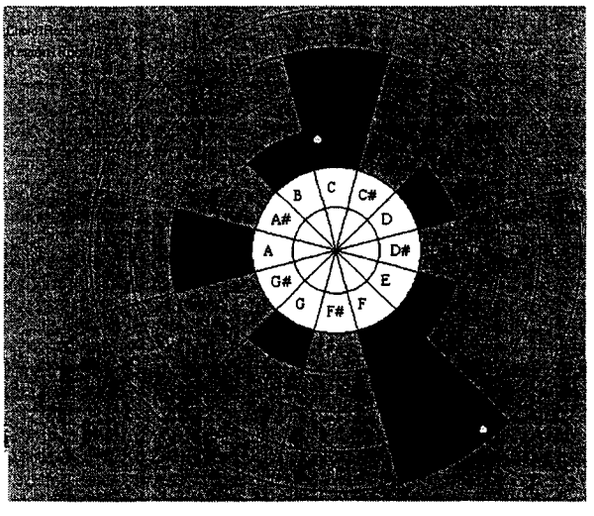


Figure 5: IV/(**I**) as F/(C)

5 2 Regions for which a chord will fit

Another question we might ask a student might be given a chord, in which regions the chord would share all three notes.

For example if, once again, the musical novice is presented with the system set up with Cmaj chord in the region Cmaj they could be asked the following question:

In addition to the region Cmaj what are the other regions in which the chord Cmaj is likely to sound nice?

Rotating the diatonic region circle clockwise from Cmaj, the student would first come to C#maj (see Figure 4). Clearly the chord Cmaj only shares one note with the region C#maj (note C, see Figure 6).

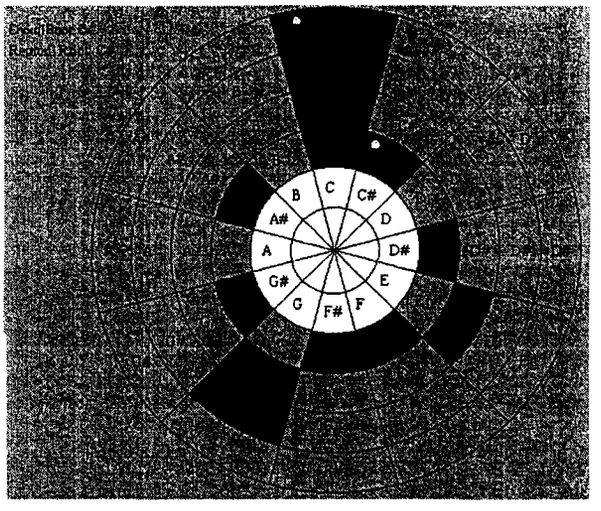


Figure 6: C/(C#)

Continuing to step the region around this way the student will come across the 2 diatonic regions in which chord Cmaj does shall all notes — region Fmaj (see Figure 7) and region Gmaj (see Figure 8).

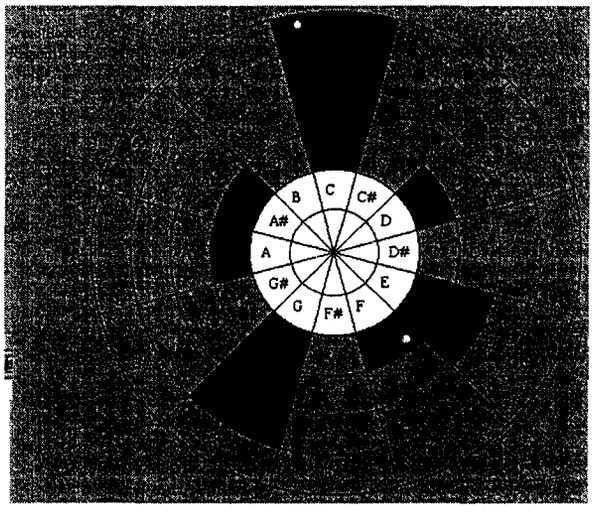


Figure 7: C/(F)

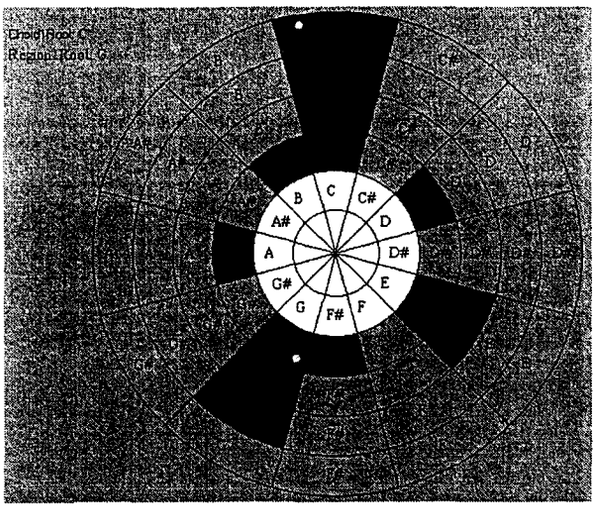


Figure 8: C/(G)

5 3 Regions that only different by one pitch class

In a similar fashion we could ask the student to derive the cycle of fifths:

Which are the closest regions to Cmaj — i.e. which regions shall all but one pitch class.

6 Conclusions & Further work

Initial, informal experiments with musical novices have been encouraging. Clearly the extended circular model maintains the features and strengths of Lerdahl's original, tabular pitch space model. Once the prototype implementation is complete stand along and comparative experiments as suggested above will be conducted. The fact that the model and computer program represent the asymmetry of the diatonic system may be important to help students move more easily from theory to practice on physical instruments where such asymmetry is unavoidable.

Extensions to the prototype being worked on include the following:

- A unified way to enable users to step and skip at any level
- The development of the tool to allow two (or more) different chords and/or regions to be visually compared (so to provide a graphical representation of Lerdahl's metrics)
- (related to above) the plotting of melodies on a circle, whereby users can choose a new set of circles each time a chord or region changes
- at present it is not possible to change the default that pitch class 0 is associated with C — although a trivial change it will enable more sophisticated and region-independent testing while still allowing students to refer to note letters if they wish

References

G. J. Balzano, The pitch set as a level of description for studying musical pitch perception, In *Music, Mind and Brain: the neuropsychology of music*, M. Clynes (Ed.), Plenum, New York, USA, 1982.

D. Deutsch, The processing of pitch combinations, in D. Deutsch (Ed.): *The Psychology of Music*, Academic Press, NY, USA, 1982.

D. Deutsch & J. Feroe, The internal representation of pitch sequences in tonal music, *Psychological Review*, 88:503-522, 1984.

S. Holland, *Artificial Intelligence, Education and Music*, Unpublished PhD thesis, IET, Open University, UK, 1989.

C. Krumhansl, The psychological representation of musical pitch in a tonal context, In *Cognitive Psychology*, 11:346-374, 1979.

C. Krumhansl, Perceptual structures for tonal music, In *Perception*, 1(1):28-62, 1983.

C. Krumhansl, J. J. Bharucha & E. Kessler, Perceived harmonic structure of chords in three related musical keys, In *Journal of Experimental Psychology: Human Perception and Performance*, 8:24-36, 1982.

C. Krumhansl & E. Kessler, Tracing the dynamic changes in perceived tonal organisation in a spatial representation of musical keys, In *Psychological Review*, 89:334-368, 1982.

C. Krumhansl & R. Shepard, *Quantification of the hierarchy of tonal functions within a diatonic context*, Presented at the Conference on Music and the Cognitive Sciences, 17-21 September, Cambridge, UK, 1979.

Fred Lerdahl, Tonal Pitch Space, *Music Perception*, 5 (3):351-350, 1988.

H. Christopher Longuet-Higgins, Two letters to a musical friend, In *The Music Review*, November 1962, 23: 244-228 & 271-280, 1982.

Arnold Schoenberg, *Theory of Harmony*, originally published 1911. Translated by R. Carter, University of California Press, Berkly, CA, USA, 1978.

R. N. Shepard, *Mental images and their transformations,* The MIT Press, Cambridge, MA, USA, 1982.

Ben Shneiderman, The future of interact systems and the emergence of direct manipulation, In *Behaviour and Information Technology*, 1:237-256, 1982.

G. Weber, *Versuch einer Geordeneten Theorie*, Mainz: B. Schotts Sohne, 1824.

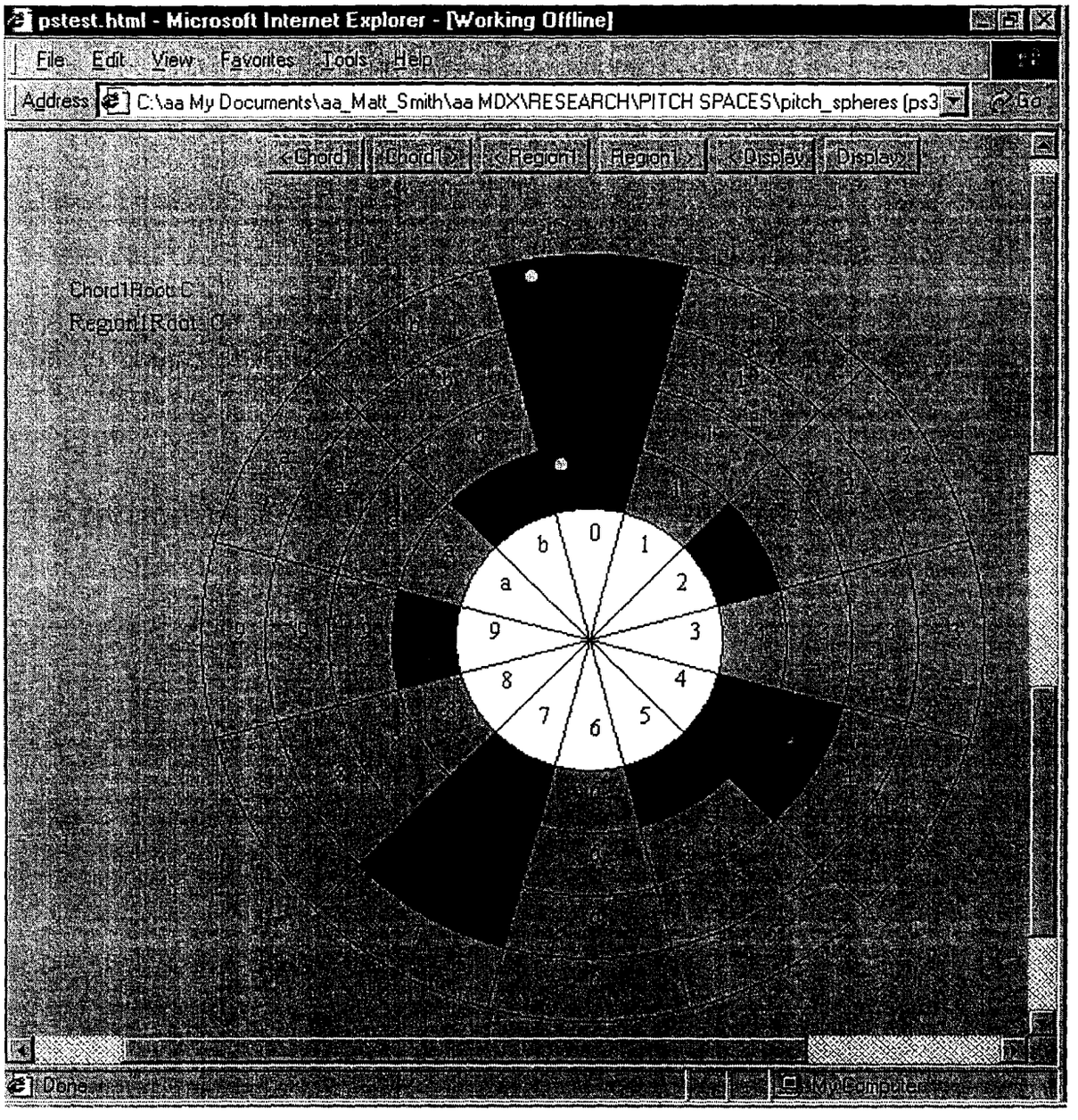


Figure 9: Prototype tool in web browser window, numeric notation mode — I/(**I**)

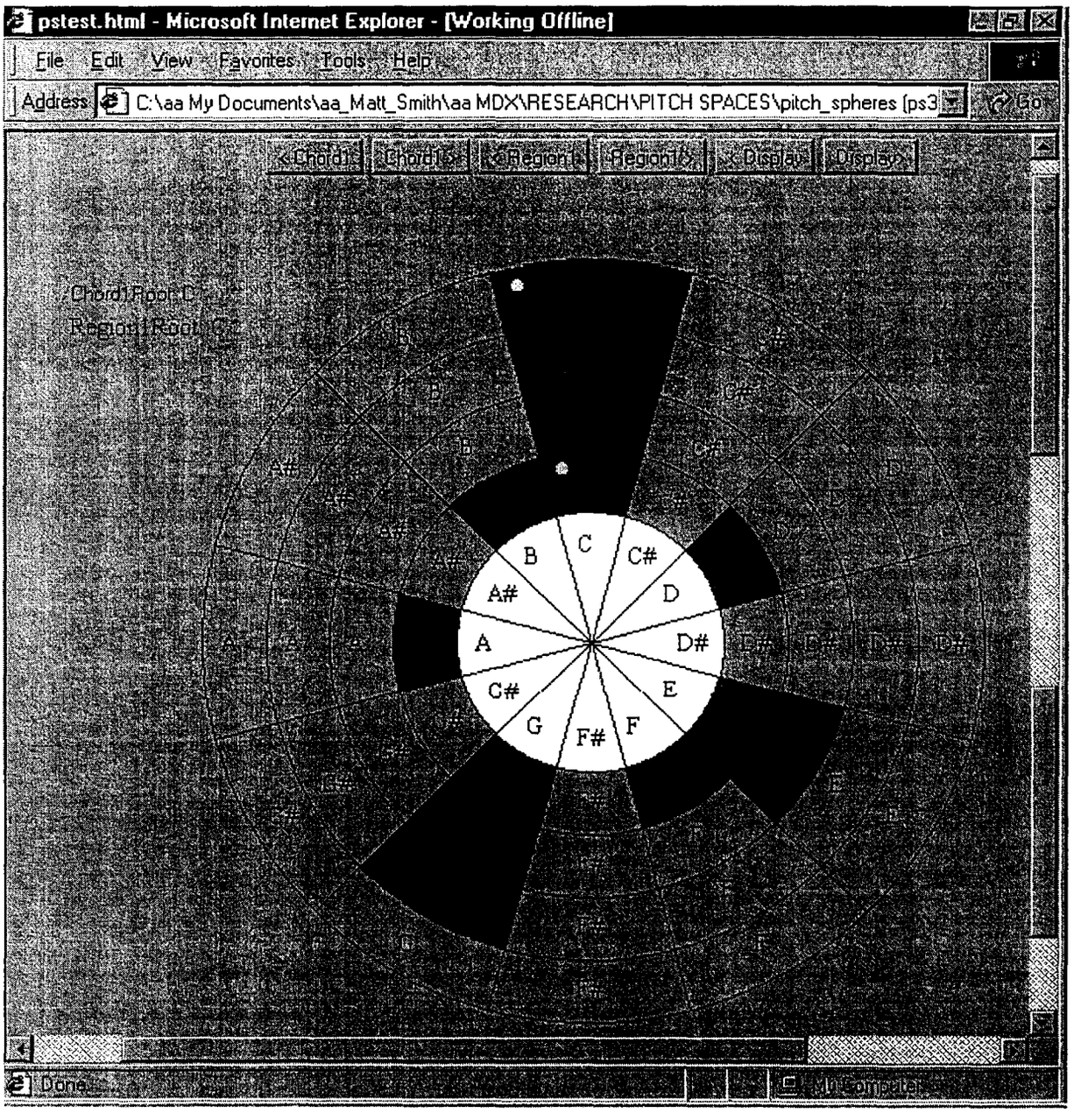


Figure 10: Prototype tool in note letter mode — I/(**I**) as C/(**C**)

Describing Verbally Expressed Humour

Graeme Ritchie
Division of Informatics
University of Edinburgh
80 South Bridge
Edinburgh
Scotland EH1 1HN

g.d.ritchie@ed.ac.uk

Abstract

In pursuit of the long-term goal of developing a general theory of humour, it is reasonable to study certain limited forms of humorous artefact in detail. One obvious class of humour to consider is verbally expressed humour, and in particular jokes. We propose a methodology for exploring this subarea. The central idea is to devise detailed symbolic descriptions of the internal linguistic structure of classes of jokes, at a suitable level of abstraction. These descriptions are intended to make explicit the semantic and pragmatic factors (broadly interpreted) that are relevant to the humorous effect of the subclass of joke in question, and also to contribute an accumulation of analysed data over which more general theorising may occur. An analogy is drawn with established practice in linguistics.

1 Motivation

The ability to comprehend, appreciate and produce humorous artefacts such as jokes is central to human culture and social interaction, and hence the area of humour (including both humorous activities and the artefacts involved) merits scientific study. Exploring humorous activity and objects in rigorous detail may throw light on (and interact with the study of) a variety of aspects of human behaviour, such as cognition, physiology, social conventions, means of communication. Artificial intelligence is well placed to pursue such an enquiry, as its methodologies and techniques have been developed to assist with the detailed symbolic modelling of complex human behaviour. The work here is an attempt to lay the foundations of such an investigation.

Although there is no accepted theory of humour, there have been numerous observations and proposals regarding the nature of humour, and these are often clustered into a tripartite division of *incongruity*, *superiority*, and *relief* "theories" (Raskin, 1985). Attardo (1994) generalises these labels to *cognitive*, *social*, and *psychoanalytical* (Figure 1). Another perspective would be to say that cognitive/incongruity approaches concentrate on the humorous *stimulus*, social/hostility approaches consider the *interpersonal* effects, and psychoanalytical/relief proposals emphasise the *audience's reaction*. All of these are interesting and valid, but distinct, aspects of the phenomenon of humour. If we are develop a complete theory of humour and its use, all of these facets must be considered. In particular, it will be necessary to have a good account of *the types of humorous stimulus that exist, how they are structured and how they function*. It is this question – the nature of the stimulus – that is addressed here. This does not constitute a complete theory of humour, but it is certainly a necessary step towards a full investigation, since it would be difficult to seek empirical support for a theory of humour use without some properly detailed analysis of the data. If we are, for example, to find correlations between types of stimuli and human reactions, then we require an account of the stimuli, to structure and guide our experimentation. The work here makes a start on dissecting the humorous stimuli, in one particular subarea of humour, namely *verbally expressed* humour. The term *verbal humour* (Raskin, 1985) is avoided here, as it is sometimes used in a narrower sense, roughly denoting plays on words (Attardo, 1994).

That is, the focus is restricted to humour conveyed in language, as opposed to physical or visual humour, but not necessarily playing on the form of the language.

Cognitive	Social	Psychoanalytical
Incongruity	Hostility	Release
Contrast	Aggression	Sublimation
	Superiority	Liberation
	Triumph	Economy
	Derision	
	Disparagement	

Figure 1: Types of "theory" (Attardo, 1994)

This restriction makes the task slightly more manageable, while still leaving a wide and rich range of phenomena to be considered.

A further simplification is choosing to study individual *jokes* in isolation from the context of use or the speaker/hearer involved. (Jokes could be loosely defined as short texts deliberately designed to elicit humorous response, often in a manner unrelated to a specific context; however, nothing here depends on having an initial definition of jokes in general.) Starting by tackling jokes out of context does not embody a claim that humorous effects are not dependent on factors such as context, personal opinions, and culture. Rather, it is an attempt to make some progress by (at least initially) not attempting to study simultaneously all the factors involved in a complex phenomenon. There is also a deliberate claim here that there will be regularities in the stimuli involved in humour (the joke texts) which should be documented and described before we can proceed to correlate these with anything else, or to devise more elaborate hypotheses about the whole humour mechanism. This is again analogous to descriptive linguistics, in which there has been a great deal of effort devoted to analysing the structural properties of *sentences*, which are in a sense the counterpart of jokes here. The hope is that if we can develop a good account of how jokes operate, then we can proceed to apply a similar methodology to other forms of verbally expressed humour.

It is important to note at the outset that this paper does *not* propose a theory of humour, not even of verbally expressed humour, nor even of jokes. What it does is to outline a methodology for approaching the construction of a theory of jokes, thence a theory of verbally expressed humour, and eventually a theory of humour. Of course, any methodological approach implicitly embodies some theoretical hypotheses, but the assumptions adopted here are relatively minimal (see Section 12 below).

2 Description and theory

If we are to develop a general theory of verbally expressed humour, it must be based on data. However, it is not feasible to proceed from raw data (e.g. a large and unanalysed collection of jokes) to a complete theory in one step. Some preliminary analysis of the data is required first. An analogy can be made here with generative linguistics, where (within the Chomskyan paradigm at least) there is a quest for a theory of universal grammar. Linguists rely, in their theory development, not on undifferentiated and unannotated data (sentences). Instead, universal theories build on previous analyses of language: fragments of grammar, or comparisons of particular constructions across languages. A large amount of pre-theoretical sorting out and dissecting of the data must occur before proceeding to highly abstract and over-arching theories of language. The position taken here is that humour research can usefully proceed in an analogous fashion (Figure 2).

Linguistics	Humour research
Strings	Grammatical texts
Sentences	Jokes
Grammaticality	Being a joke
Sentence type	Subclass of jokes
Structural description	Description of a joke
Grammar rule(s)	Pattern for class of joke
Theory of grammar	Theory of jokes
Theory of language use	Theory of humour use

Figure 2: Analogy with linguistics

Before we can construct a genuinely empirical general theory of (verbally expressed) humour, we must carry out a significant amount of groundwork which involves analysing our primary data. Below, we make some suggestions about how such descriptive work could proceed, with some illustrative analyses of simple jokes.

3 What is an analysis?

The overall idea behind the framework here is simple: in order to make clear and explicit the various factors that contribute (or might contribute) towards the humorous effect of a piece of text, one should specify in some detail the various abstract objects that are posited as underlying the texts (e.g. symbolic representations of meaning), the various properties that these objects have (e.g. denoting a taboo subject) and the various interrelations which hold between them (e.g. one meaning being more obvious than another, one word sounding similar to another). An analysis of a joke is then a precise listing of this information for the joke, at a suitable level of abstraction. This last point is important: we have to rely (at least at this stage of methodological development) on the intuition and judgement of the analyst to propose abstract entities which are relevant to the humorous effect, while ignoring irrelevant details. For the moment, the assessment of the suitability of the components of the joke analysis will be left to the informed opinion of other humour researchers, but in time we should evolve a more sophisticated methodology in which independent criteria can be brought to bear. Appealing again to the analogy with generative linguistics, when a syntactic rule is posited for a particular class of sentences, there are various non-subjective criteria that can be applied to argue for or against the adequacy or elegance of the rule. In describing verbally expressed humour, analogous criteria must be developed, for example by arguing that the joke analysis offered will delineate some natural class of jokes (cf. Ruch et al. (1993)). Classes of joke can then be characterised by abstracting from individual descriptions to form more general patterns, in a manner analogous to creating grammar rules which define the structure of classes of sentences.

4 Basic objects

We shall start from a few basic data types, and build from there, introducing new primitives only as particular types of joke seem to demand them.

Most discussions of jokes do not make it explicit what their assumed primitive alphabet is. Since jokes are conveyed sometimes in speech and sometimes in writing, either phonetics or orthography could be chosen. For many jokes, the choice is immaterial. For some jokes, the spoken delivery is essential in order to create some form of ambiguity. Only very rarely is it necessary to use a written form in order to create the desired effect. In general, it is up to the analyst to define which alphabet is to be the formal representation for a joke, but here we shall assume that there is such an alphabet, and that any string over that alphabet constitutes a *text*. That is, we will use the technical term "text" to cover any sequence of symbols, whether a complete joke (or well-formed sentence) or not. We shall also assume that there is a *similarity measure* which indicates how similar two texts are, in some primitive sense. (identity of texts will simply be identity within the set of strings over the alphabet). This will be useful in analysing jokes involving puns or ambiguity. It will then be possible to define various kinds of near-equality relation between texts, based on degrees of similarity (cf. the "paraphony" and "hahaphony" of Dienhart (1999)).

Our initial set of object-types is then:

(a) ALPHABET: a set of basic symbols from which jokes are (at the simplest level) made up; the analyst should make it clear whether written or spoken symbols are intended.

(b) TEXT: a TEXT is a sequence of elements from the chosen alphabet. Hence any substring of a TEXT is a TEXT. There is a *similarity measure* which indicates how similar two TEXTs are.

(c) MEANING: a MEANING is what might be termed the *literal meaning* or *semantic structure* of a TEXT. It takes no account of any inference or contextual information which might flesh out or interpret the meaning of the actual words used.

(d) INTERPRETATION: A TEXT may also have an associated INTERPRETATION, which will be dependent upon (but not identical to) the MEANING of the TEXT (or its parts). It can be thought of as an interrelated and consistent set of propositions, with more content than the bare MEANINGs. This is intended to be a broader kind of meaning (of a passage of text or of some sequence of events, for example), which may involve much reasoning, filling in of implicit information, unwarranted addition of assumptions, etc.

(e) DESCRIPTION: This is a semantic structure which encodes some attributes which could be true of an entity; it can *describe* a MEANING.

These classes of object are primitive in the sense that they are defined solely in terms of the relationships they enter into with other primitive objects. MEANINGs, INTERPRETATIONs and DESCRIPTIONs are all SEMANTIC ITEMS, sometimes abbreviated below to "*SemItem*". All of the above entities are linguistic or abstract representational forms. There will also be entities denoting objects or situations within some world (real or imaginary); see Section 10 for two simple examples.

5 Properties and relations

In addition to some directly linguistic relationships between our basic objects, there will be a large set of attributes which are relevant to recording the humorous mechanisms within a joke. Thus our conceptual repertoire will range from the relatively straightforward (e.g. one TEXT is a substring of another), through conventional linguistic notions (e.g. a TEXT may have zero or more MEANINGs), to quite difficult and non-trivial properties (e.g. a MEANING is absurd, or an INTERPRETATION conveys a taboo idea). This paper will *not* attempt to give full and precise definitions of these predicates, although we will provide informal glosses of those which we use, in order to make the examples intelligible. Supplying detailed definitions for all the predicates involved in descriptions of jokes constitutes a central and substantive part of developing a theory of jokes, and thence a theory of humour (cf. (Ritchie, 1999, Section 4.5)). The first step in our methodology is to postulate a range of these constructs, and see if we can account for joke structure by using them consistently. This decomposes the research into stages, with the final definitions of these conceptual building blocks being postponed until we have an idea of the set of primitives that we need.

Two illustrative and typical examples might be as follows:

$absurd(\langle SemItem \rangle)$: This is true if $\langle SemItem \rangle$, a SEMANTIC ITEM, is in some way odd or bizarre.

$conflicts(\langle Meaning \rangle, \langle SemItem \rangle)$: $\langle Meaning \rangle$ will not merge with $\langle SemItem \rangle$ to form a coherent INTERPRETATION.

6 Structural descriptions

To set out a description of the linguistic content of a joke, we need to state exactly what abstract objects we are positing and what the relationships are between them. Many authors have remarked on the way in which certain jokes use the final line (punchline) to reveal an unexpected meaning for the initial text, in a way that implies, evokes, or describes an image that is odd in some way ((Raskin, 1985), (Attardo, 1994, Chapter 2), (Ritchie, 1999)). To illustrate the descriptive approach here, we can borrow

this informal idea and state it in our terminology. (Notice that the examples in this paper are mostly chosen for their brevity and simplicity rather than the excellence of their wit.)

(1) Why do birds fly south in winter?
It's too far to walk.

The relevant workings of this joke could be summarised thus:

> There is a TEXT T = "Why do birds fly south in winter? It's too far to walk." There are subsequences T_1 = "Why do birds fly south in winter?", T_2 = "It's too far to walk", and MEANINGs M_1, M_2, M_3 such that M_1 is a more **obvious** MEANING than M_2 for T_1, M_3 is the MEANING of T_2, M_3 **conflicts** with M_1 (or perhaps an INTERPRETATION derived from M_1), M_3 is **compatible** with M_2, and there is an INTERPRETATION I of M_2+M_3 which is **absurd**.

(where words in **bold** font indicate properties or relations which this description relies on).

7 Structural patterns

The above summary describes the content of one particular example joke. However, a necessary next step is to abstract from such itemisations to create more general patterns, which will describe classes of jokes which are similar in their internal workings. Here we will call these *structural patterns*. For the sake of a simple notation, we will adopt a sorted version of first order predicate logic (FOPL).

The above example can be seen as an instance of a broader class characterised by the following (where T is the text of the joke):

$$\begin{array}{l}\exists T_1, T_2 : text; M_1, M_2, M_3 : meaning; \\ I : interpretation \\ such\ that \\ substrings(T, T_1, T_2) \wedge meaning(T_2, M_3) \\ \wedge\ obviousmeaning(T_1, M_1, M_2)\ \wedge \\ conflicts(M_3, M_1) \wedge compatible(M_3, M_2) \wedge \\ form\_interpretation(M_2, M_3, I) \wedge absurd(I)\end{array}$$

Here we have introduced further predicates:

$obviousmeaning(\langle Text\rangle, \langle Sem_1\rangle, \langle Sem_2\rangle)$: This is true if $\langle Sem_1\rangle$ is a more obvious interpretation of $\langle Text\rangle$ than $\langle Sem_2\rangle$.

$compatible(\langle Meaning\rangle, \langle SemItem\rangle)$: $\langle Meaning\rangle$ can merge with $\langle SemItem\rangle$ to form a coherent INTERPRETATION.

$form\_interpretation(\langle Sem_1\rangle, \langle Sem_2\rangle, \langle Sem_3\rangle)$: $\langle Sem_1\rangle$ merged with $\langle Sem_2\rangle$ forms $\langle Sem_3\rangle$.

8 Notation

Although we have adopted FOPL as our notation for expressing the information about a joke, this in no way implies that the content of jokes is "logical" in any ordinary sense of the word; the FOPL notation is merely acting as our metalanguage: a convenient and concise way of writing down statements which involve abstract *objects*, *properties* and *relations* (see the summary of assumptions in Section 12 below). The use of a logical representation should facilitate the detection of classes and subclasses of jokes, as such inclusions (or similarities) will be reflected in logical expressions which subsume or overlap with one another.

FOPL has its merits as a meta-notation, but it could be rather verbose if used to state structural patterns in the manner shown above. For example, the existence of substrings has to be stated every time, and will not usually be very complex, so it may not be necessary to have the full expressive power of logic merely to state this information. Also, patterns will always be of a form in which some existentially quantified variables are introduced, and then some constraints are placed on them. Moreover, there are some recurring interrelations (e.g. that a meaning is associated with a particular stretch of text). Therefore, it is worthwhile developing a more succinct and perspicuous notation (which could in principle be expanded into FOPL).

Firstly, we can indicate the segmentation of the text into parts using brackets and subscripts:

> $[_1$Why do birds fly south in winter?$]_1$
> $[_2$It's too far to walk.$]_2$

The subscripts can be used in terms indicating the various objects involved, so that $\mathcal{M}(1)$ is the MEANING of TEXT labelled 1 (which in turn is notated as $\mathcal{T}(1)$). Where there are more than one possible MEANING for a particular TEXT $\mathcal{T}(N)$, these will be indicated by $\mathcal{M}(N_a)$, $\mathcal{M}(N_b)$, etc. An INTERPRETATION formed from the MEANINGs $\mathcal{M}(1), \ldots \mathcal{M}(n)$ will be written $\mathcal{I}(1, \ldots, n)$. Also, we can assume that all variables mentioned (or implicitly used) in the logical expressions are existentially quantified. The properties of items, and relationships between items, can still be written using the notation of FOPL. The example description can then be given as:

$$\begin{array}{l}obviousmeaning(\mathcal{T}(1), \mathcal{M}(1_a), \mathcal{M}(1_b)) \wedge \\ conflicts(\mathcal{M}(2), \mathcal{M}(1_a))\ \wedge \\ compatible(\mathcal{M}(2), \mathcal{M}(1_b)) \wedge absurd(\mathcal{I}(1_b, 2))\end{array}$$

This has the merits of building the more basic linguistic relations into the notation, thus rendering the more substantive and humour-related predicates more prominent. That is, the notation is *not* being used to express interesting theoretical claims (as is sometimes advocated in early Chomskyan linguistics); instead, it is introduced to push the less interesting structure into the background and let the potentially significant predicates appear clearly.

To extend this notation to structural patterns as well as descriptions of individual jokes, we can indicate the decomposition of the text into substrings by a diagram of labelled brackets showing the relative positioning of the segments. That is, each pattern will have a "header" which shows the shape of the text, numbering its subparts; for example:

$[_1 \ldots]_1[_2 \ldots]_2$

indicates a text made up of two substrings. The above example (1) is then an instance of the following pattern:

$[_1 \ldots]_1[_2 \ldots]_2$
$obviousmeaning(\mathcal{T}(1), \mathcal{M}(1_a), \mathcal{M}(1_b)) \wedge$
$conflicts(\mathcal{M}(2), \mathcal{M}(1_a)) \wedge$
$compatible(\mathcal{M}(2), \mathcal{M}(1_b)) \wedge absurd(\mathcal{I}(1_b, 2))$

In this case, the application of the pattern to the specific example is encoded entirely in the binding of the text segments 1, 2, etc.

9 Descriptive jokes

Some very brief examples may help to demonstrate the approach advocated here. These make central use of the DESCRIPTION data type introduced earlier.

(2) Why is coffee like the soil?
It is ground. (Pepicello and Green, 1984)

This example could be described as:

$[_1 \ldots]_1[_2 \ldots]_2$
$compares(\mathcal{T}(1), M, N) \wedge$
$yields\_description(\mathcal{M}(2), D) \wedge$
$describes(D, M) \wedge describes(D, N)$

providing that we have the following predicates:

$yields\_description(\langle Meaning \rangle, \langle Description \rangle)$: This is true if the $\langle Description \rangle$ can be extracted from the $\langle Meaning \rangle$.

$describes(\langle Description \rangle, \langle Meaning \rangle)$: This is true if $\langle Description \rangle$ describes $\langle Meaning \rangle$.

$compares(\langle Text \rangle, \langle Meaning_1 \rangle, \langle Meaning_2 \rangle)$: This is true if $\langle Text \rangle$ implies or states that $\langle Meaning_1 \rangle$ and $\langle Meaning_2 \rangle$ are similar.

To describe example (3)

(3) What is grey, has four legs, and a trunk? A mouse on vacation. (Rothbart, 1977)

we require the following predicate:

$obviousdescription(\langle Desc \rangle, \langle Meaning_1 \rangle, \langle Meaning_2 \rangle)$: This is true if both $\langle Meaning_1 \rangle$ and $\langle Meaning_2 \rangle$ are described by the DESCRIPTION $\langle Desc \rangle$, but this is more obviously the case for $\langle Meaning_1 \rangle$ than for $\langle Meaning_2 \rangle$.

The structural pattern is then:

$[_1 \ldots]_1[_2 \ldots]_2$
$yields\_description(\mathcal{M}(1), D) \wedge absurd(\mathcal{M}(2))$
$\wedge\ obviousdescription(D, M, \mathcal{M}(2)))$

Example (4)

(4) What do you call a strange market? A bizarre bazaar. (Binsted, 1996)

can be described as:

$[_1 \ldots]_1[_2[_3 \ldots]_3[_4 \ldots]_4]_2$
$yields\_description(\mathcal{M}(1), D) \wedge$
$describes(D, \mathcal{M}(2)) \wedge soundalike(\mathcal{T}(3), \mathcal{T}(4))$

where the definition of *soundalike* can be based on the similarity metric for TEXTs.

10 Narrative jokes

A large class of more complex jokes are those which rely on narrative (i.e. "funny stories"). To describe the internal workings of such jokes, particularly those which have a "butt" or "target", we need to introduce data-types denoting entities (concrete or abstract) within the world of the story. For the moment, we will restrict ourselves to an EVENT-SEQUENCE, which a TEXT *narrates*, and a data-type CHARACTER for denotations of individuals within a story.

For example, consider (5).

(5) Russian officers in an Eastern European country go to a tavern. They order beer. The waiter places coasters on the table and serves the beer. Later they order another round. The waiter returning with the beer finds no coasters. 'OK,' he tells himself, 'these are collectors,' and puts down another set of coasters. When the third round is ordered and brought out, there are again no coasters. Angry, the waiter puts the beer down on the table, but places no more coasters. One of the Russian officers protests: 'What's this? No more crackers?' (Hetzron, 1991, p.62)

This could be approximated with the pattern:

$[_1 \ldots]_1[_2 \ldots]_2$
$narrates(\mathcal{T}(1), E) \wedge$
$obviousinterpretation(E, I_1) \wedge$
$conflicts(\mathcal{M}(2), I_1) \wedge adopts(\mathcal{I}(2), C, I_2, E) \wedge$
$different(I_1, I_2) \wedge absurd(I_2)$

assuming we use the following predicates:

$narrates(\langle Text \rangle, \langle Events \rangle)$: $\langle Text \rangle$ recounts the EVENT-SEQUENCE $\langle Events \rangle$.

$obviousinterpretation(\langle Events\rangle, \langle Interp\rangle)$: The natural INTERPRETATION for $\langle Events\rangle$ is the INTERPRETATION $\langle Interp\rangle$.

$adopts(\langle InterpA\rangle, \langle Char\rangle, \langle InterpB\rangle, \langle Events\rangle)$: In INTERPRETATION$\langle InterpA\rangle$, the CHARACTER $\langle Char\rangle$ adopts the INTERPRETATION$\langle InterpB\rangle$ for $\langle Events\rangle$.

along with some notion of "distinctness" of INTERPRETATIONs.

This emphasises the difference between MEANINGs and INTERPRETATIONs. The idea that the coasters have been consumed by the soldiers, is part of the INTERPRETATION of the narrative, although it is not stated as literal meaning. Also, the idea (imputed to the soldiers) that the coasters are crackers is an interpretation of the waiter's actions (within the world of the story), and is not a literal meaning of any fragment of text.

Although the analyses given above suggests that the relevant property of the soldiers' interpretation is that it is *absurd*, the pattern could perhaps be generalised to cover more jokes in a fairly natural way. Consider (6), the central example from Raskin (1985).

(6) 'Is the doctor at home?' the patient asked in his bronchial whisper. 'No,' the doctor's young and pretty wife whispered in reply. 'Come right in.'

This broadly follows the same pattern as (5). It could be argued that the amusing interpretation adopted by the character in this story is not so much *absurd* as *taboo* (with its implication of adultery). We could generalise the structural pattern by replacing *absurd* with *inappropriate*, which we would define to be a disjunction of *absurd*, *taboo*, and perhaps other properties found to render interpretations amusing. Notice that *some* property is essential, otherwise the stories would be simple tales of misunderstandings, with no humorous effect. Raskin argues (in keeping with his semantic script-based theory of humour) that in (6) the important ingredient is not located in the mistaken interpretation alone, but in a form of comparison (*script opposition*) with the more obvious and natural interpretation. If we were to accept this idea as being a possible ingredient in making tales of misunderstanding funny, then the last term in the above pattern could be changed from $absurd(I_2)$ to $(inappropriate(I_2) \vee contrast(I_1, I_2))$ where *contrast* embodies the appropriate form of opposition.

(Notice that we have not yet covered the satirical or mocking aspect of (5), as our structural patterns give no indication that this joke casts a slur on Russian officers. Such indirect or inferred content goes beyond the current paper.)

It is interesting to see how this pattern can also cover some apparently simpler jokes, such as (7) and (8).

(7) 'Mr Fields, do you believe in clubs for young people?' 'Only when kindness fails.' (Shultz, 1976) and elsewhere.

(8) A lady went into a clothing store and asked 'May I try on that dress in the window?' 'Well,' replied the sales clerk doubtfully, 'don't you think it would be better to use the dressing room?' (Oaks, 1994), citing from Clark (1968)

It might seem natural to focus on the fact that these examples rely on linguistic ambiguity (lexical ambiguity in (7), syntactic structural ambiguity in (8)), and to posit a pattern such as:

$$[_1 \ldots]_1 [_2 \ldots]_2$$
$$obviousmeaning(\mathcal{T}(1), \mathcal{M}(1_a), \mathcal{M}(1_b)) \wedge$$
$$conflicts(\mathcal{M}(2), \mathcal{M}(1_a)) \wedge$$
$$compatible(\mathcal{M}(2), \mathcal{M}(1_b)) \wedge absurd(\mathcal{I}(1_b, 2))$$

However, this would miss a generalisation. In these very short stories, the utterances attributed to specific characters (the questioner in (7) and the lady in (8)) are in fact events – linguistic in nature – which are being narrated, and the ending of the joke involves some other character imposing an interpretation on these linguistic events which is not the obvious interpretation, and which is inappropriate in some way. Hence they both fall under the more general pattern given for (5) (as amended to go beyond *absurd* in its last line). The linguistic ambiguity in (7) and (8) is then viewed solely as a means to an end, since it is the ambiguity which allows the different possible interpretations of the linguistic events.

Taking this perspective is different from some more traditional humour analyses (cf. (Attardo, 1994, Chapter 2)), in which a major dividing line is drawn between jokes which depend for their effect on the language in which they are expressed (*verbal* jokes) and jokes which are more easily translatable into other languages because the exact phrasing is not crucial (*referential* jokes). In such a taxonomy, (7) and (8) would be verbal, while (5) and (6) would be referential. That would obscure the generalisation, which we conjecture is a useful one, that all these four stories share a common mechanism which is significantly involved in their status as jokes.

11 Possible implementations

The work reported here is very preliminary, and does not, at present, involve computational implementation. However, the emphasis on formalisation and detail is intended to lead towards fuller symbolic models which could be implemented and tested. There are various ways in which this work could lead to implementation.

A rule tester. In the same way that a linguist could make use of a *grammar testing program* to check their rules (e.g. Friedman (1971)), software could be constructed to apply rules to data (much as was done with the JAPE system (Binsted et al., 1997)). If every output item is deemed to be a joke (by a suitable set of human subjects), our

Acknowledgements

Thanks are due to the members of the Alfa Informatica group at the University of Amsterdam, where this work was largely carried out.

References

Salvatore Attardo. *Linguistic Theories of Humour*. Mouton de Gruyter, Berlin, 1994.

Kim Binsted. *Machine humour: An implemented model of puns*. PhD thesis, University of Edinburgh, Edinburgh, Scotland, 1996.

Kim Binsted, Helen Pain, and Graeme Ritchie. Children's evaluation of computer-generated punning riddles. *Pragmatics and Cognition*, 5(2):309–358, 1997.

David Allen Clark. *Jokes, Puns and Riddles*. Doubleday, New York, 1968.

John M. Dienhart. A linguistic look at riddles. *Journal of Pragmatics*, 31(1):95–125, 1999.

Joyce Friedman. *A Mathematical Model of Transformational Grammar*. American Elsevier, New York, 1971.

Robert Hetzron. On the structure of punchlines. *HUMOR*, 4(1):61–108, 1991.

Dallin D. Oaks. Creating structural ambiguities in humor: getting English grammar to cooperate. *HUMOR*, 7(4): 377–401, 1994.

William J. Pepicello and Thomas A. Green. *The Language of Riddles*. Ohio State University Press, Columbus, Ohio, 1984.

Victor Raskin. *Semantic Mechanisms of Humour*. Reidel, Dordrecht, 1985.

Graeme Ritchie. Developing the incongruity-resolution theory. In *Proceedings of the AISB Symposium on Creative Language: Stories and Humour*, pages 78–85, Edinburgh, Scotland, 1999.

Mary K. Rothbart. Psychological approaches to the study of humour. In Anthony J. Chapman and Hugh C. Foot, editors, *It's a Funny Thing, Humour*, pages 87–94. Pergamon Press, Oxford, 1977.

Willibald Ruch, Salvatore Attardo, and Victor Raskin. Toward an empirical verification of the general theory of verbal humour. *HUMOR*, 6(2):123–136, 1993.

Thomas R. Shultz. A Cognitive-Developmental Analysis of Humour. In Anthony J. Chapman and Hugh C. Foot, editors, *Humour and Laughter : Theory, Research and Applications*, chapter 1, pages 11–36. Transaction Publishers, London, first edition, 1976.

Towards A Computational Model of Poetry Generation

Hisar Maruli Manurung, Graeme Ritchie, Henry Thompson
Division of Informatics
University of Edinburgh
80 South Bridge Edinburgh EH1 1HN
hisarm@dai.ed.ac.uk, g.d.ritchie@ed.ac.uk, ht@cogsci.ed.ac.uk

Abstract

In this paper we describe the difficulties of poetry generation, particularly in contrast to traditional *informative* natural language generation. We then point out deficiencies of previous attempts at poetry generation, and propose a stochastic hillclimbing search model which addresses these deficiencies. We present both conceptual and implemented details of the most important aspects of such a model, the evaluation and evolution functions. Finally, we report and discuss results of our preliminary implementation work.

1 Motivation

Poetry is a unique artifact of human natural language production, with the distinctive feature of having a strong unity between its content and its form. The creation of poetry is a task that requires intelligence, expert mastery over world and linguistic knowledge, and creativity. Although some research work has been devoted towards creative language such as story generation, poetry writing has not been afforded the same attention. It is the aim of this research to fill that gap, and to shed some light on what often seems to be the most enigmatic and mysterious forms of artistic expression.

Furthermore, poetry possesses certain characteristics that render traditional natural language generation (NLG) systems, which are geared towards a strictly *informative* goal, unsuitable due to architectural rigidness.

Lastly, although not readily obvious, there are potential applications for computer generated poetry, such as the increasingly large industry of electronic entertainment and interactive fiction, the commercial greeting card poetry genre, and perhaps even the odd pop music lyric or two.

2 Poetry Generation

2.1 What Is Poetry?

Regarding poetry the artifact, Levin (1962) states that "In poetry the form of the discourse and its meaning are fused into a higher unity." This definition highlights the point of a strong interaction between semantics, syntax and lexis. Boulton (1982) reiterates this point, claiming that it is misleading to separate the physical and intellectual form of a poem so far as to ask, "What does it mean?". The poem means itself.

These are rather esoteric quotations, and one would expect these quotes to be referring to "high-brow" poetry. However, we claim that this unity is inherent in simpler forms of poetry, for example, Hillaire Belloc's *The Lion* (Daly, 1984):

The Lion, the Lion, he dwells in the waste,
He has a big head and a very small waist;
But his shoulders are stark, and his jaws they are grim,
And a good little child will not play with him.

Essentially unity here means that the poem "works" due to a combination of features at the surface level (the rhyming of *waste* and *waist*, *grim* and *him*, the repetition of *The lion, the lion* to fit the rhythm), and semantics (description of a lion as told to a child).

Of the many special characteristics that poetic form possesses, among the most essential are: rhythm, rhyme, and figurative language.

As for the process of writing poetry, it is often claimed to proceed in a much more flexible manner than other writing processes. There is often no well-defined communicative goal, save for a few vague concepts such as "wintery weather" or "a scary lion". Furthermore, a human could begin writing a poem inspired by a particular concept, or scenario, but end up writing a poem about an altogether different topic.

This specification of loose constraints fits with (Sharples, 1996) and (Boden, 1990), who claim that while a writer needs to accept the constraints of goals, plans, and schemas, *creative* writing requires the breaking of these constraints. Yet these constraints are still necessary, as they allow for the recognition and exploiting of opportunities.

Sharples (1996) models the writing process as that of creative design, involving a cycle of analysis, known as *reflection*, and synthesis, known as *engagement*. This pro-

cess is analogous to our iterative process of evaluation and evolution, and ties in with the concept of unity between content and form: during the reflection phase, when looking at an intermediate draft of the poem on paper, a poet may come to realize the opportunities of surface features that can be exploited, which enables further content to be explored upon subsequent engagement phases.

2.2 Previous Attempts

Most previous attempts at poetry generation are "hobbyist experiments" that are available on the World Wide Web, such as The Poetry Creator, ELUAR, and Pujangga, with the two exceptions that exist in publication being RACTER and PROSE (Hartman, 1996). RACTER is also the only computer program with a published poetry anthology, "The Policeman's Beard is Half Constructed" in 1984.

All of these attempts were essentially "party trick"-type programs, in the mould of ELIZA (Weizenbaum, 1966). Typically, the generation process simply consisted of randomly choosing words from a hand-crafted lexicon to fill in the gaps provided by a template-based grammar. However, several clever tricks and heuristics were employed on top of the randomness to give the appearance of coherence and poeticness, such as: (1) assigning ad-hoc "emotional categories", e.g. {ethereality, philosophy, nature, love, dynamism} in ELUAR, and {romantic, patriotic, wacky, moderate} in Pujangga, (2) choosing lexical items repetitively to give a false sense of coherence, e.g. RACTER, (3) constructing highly elaborate sentence templates, often to the point that the resulting poetry would have to be attributed more to the human writer than to the program.

This is a representative output from ELUAR:

> *Sparkles of whiteness fly in my eyes,*
> *The moan of stars swang branches of trees,*
> *The heart of time sings in the snowy night.*
> *Seconds of Eternity fly in grass,*
> *The Clock of rain turns,*
> *Death of the Apples,*
> *The Equinox penetrates the words.*

The two great deficiencies of these attempts were that they took no account whatsoever of semantics (the systems were not trying to convey any message) nor of poetic form, e.g. rhythm, rhyme, and figurative language.

2.3 What Makes It Difficult?

If we are to develop a poetry generation system which overcomes the two deficiencies mentioned above, what difficulties do we run into, particularly in comparison to conventional NLG systems?

1. In conventional, informative NLG systems, the starting point is a given message, or communicative goal, and the goal is to produce a string of text that conveys that message according to the linguistic resources available. In poetry, however, there may not be a well-defined message to be conveyed (see section 2.1).

2. The generation process is commonly decomposed into stages of content determination, text planning, and surface realisation (Reiter, 1994). We claim this approach is unsuitable for the task of poetry generation because it introduces problems of architectural rigidness (cf. De Smedt et al. (1996)) which are exacerbated by the unity of poetry, where interdependencies between semantics, syntax, and lexis are at their strongest.

3. If our poetry generator is to create texts which satisfy the multitude of phonetic, syntactic and semantic constraints, it must have a very rich supply of resources, namely: a wide coverage grammar which allows for paraphrasing, a rich lexicon which supplies phonetic information, and a knowledge-base if we hope to produce coherent poems.

4. One of the main difficulties lies in the objective evaluation of the output text. The question of measuring text quality arises for existing NLG systems, but is much more pronounced in evaluating poetry: how does one objectively evaluate if something is a poem or not.

   When writing about Masterman's haiku producer, Boden (1990) states that readers of poetry are prepared to do considerable interpretative work, and the more the audience is prepared to contribute in responding to a work of art, the more chance there is that a computer's performance may be acknowledged as aesthetically valuable. Hence readers of computer-generated text will be more tolerant in their assessment of poetry than of prose.

   This sounds encouraging for doing work in poetry generation. However, this observation also implies that it could be *too* easy to program the computer production of poetry: precisely because poetry readers are prepared to do interpretative work, it would be all too easy to pass off random word-salad output as Truly Genuine Poetry (whatever that may be).

The first three points mentioned above are of a more technical nature, while the last one is more conceptual, perhaps even philosophical. Currently, we do not have much to say on this last point, except that we hope to adopt an objective and empirical evaluation methodology similar to that of Binsted et al. (1997).

2.4 Limiting our Poetry

The main characteristics which we look for in our target generated poetry are a highly regular occurrence of syn-

tactic and phonetic patterns, such as metre, rhyme, and alliteration. These are easily identifiable, and one could say we are adopting a "classic" view of poetry. Furthermore, we will only offer a relatively simple and straightforward treatment of semantics (see section 4.3) and of constructing the poem's content. The verse in section 2.1 typifies these attributes.

3 A Stochastic Hillclimbing Model

In an attempt to address the difficulties raised in Section 2.3, we propose to model poetry generation as an explicit search, where a state in the search space is a possible text with all its underlying representation, and a "move" in the space can occur at any level of representation, from semantics all the way down to phonetics. This blurs the conventional divisions of content determination, text planning, and surface realisation, and is actually readopting what De Smedt et al. (1996) call an *integrated architecture*, which goes against recent developments in NLG, but seems a necessary decision when considering poetry.

The problem, of course, is navigating the prohibitively large search space. Our proposed solution is to employ a stochastic hillclimbing search, not merely for its relatively efficient performance, but especially since the creative element of poetry generation seems perfectly suited to a process with some element of randomness to it.

Our stochastic hillclimbing search model is an *evolutionary* algorithm, which is basically an iteration of two phases, *evaluation* and *evolution*, applied to an ordered set (the *population*) of candidate solutions (the *individuals*).

This approach is quite analogous to (Mellish et al., 1998), an experiment in using stochastic search for text planning, but in our research we extend it to the whole NLG process.

3.1 Evaluation

Arguably the most crucial aspect of a stochastic search is the evaluation scheme which lets the system know what a desirable solution is. Below we present an informal description, not necessarily exhaustive, of the features that our evaluation functions must look for in a poem. A description of the actual evaluators in our currently implemented system can be found in Section 4.5.

1. **Phonetics**: One of the most obvious things to look for in a poem is the presence of a regular phonetic form, i.e. rhyme, metre, alliteration, etc. This information can be derived from a pronunciation dictionary.

   One possible evaluation method is to specify a "**target phonetic form**" as input, i.e. the ideal phonetic form that a candidate solution should possess, and to then score a candidate solution based on how closely it matches the target form.

   For example, we could provide the system with the following target form (here `w` means a syllable with weak stress, `s` a syllable with strong stress, and `(a)` and `(b)` would determine the rhyme scheme, e.g. *aabba*), which effectively means we are requesting it to generate a *limerick*:

   ```
   w,s,w,w,s,w,w,s(a)
   w,s,w,w,s,w,w,s(a)
      w,s,w,w,s(b)
      w,s,w,w,s(b)
   w,s,w,w,s,w,w,s(a)
   ```

   Alternatively, we could specify a set of these target forms, thus feeding the system with knowledge of existing poetry forms: the quintain, haiku, rondeau, sestina, etc., and allow the evaluation function to reward candidate solutions that are found gravitating closely towards one of those patterns. This is a more flexible alternative, but would probably not be as informed a heuristic, as the definition of the goal becomes less focussed.

2. **Linguistics**: Aside from phonetic patterns, there are other, more subtle, features to look for in a poem: **lexical choice**, where the evaluation could reward the usage of interesting collocations and words marked as "poetic", **syntax**, where reward could be given to usage of interesting syntactic constructs, e.g. inverse word and clause order, topicalization, and **rhetoric**, where evaluation would score the usage of figurative language constructs such as metonymy.

3. **Semantics**: Even more abstract would be a mechanism for evaluating the semantics of a certain candidate. Again, we could specify a "target semantics" and score a candidate's semantics relative to this target. Unlike conventional NLG, though, this target semantics is not viewed as a message that must be conveyed, but rather as a "pool of ideas", from which the system can draw inspiration. The system could choose to convey more or less than the given semantics (cf. *approximate generation* in Nicolov (1998)).

   Story generation issues such as narrative structure and interestingness are beyond the scope of this research.

Having analysed the three points above, it seems that to devise an evaluation function, the following 3 issues must be tackled:

- How to **identify** the presence of a feature: with the possible exception of figurative language, it is reasonably straightforward to observe the features. Most of them are represented directly in the data structure, e.g. phonetic form, lexical choice, syntactic structure, semantic interpretation.

- How to **quantify** a feature: yielding a numerical measure for the occurrence of a poetic feature sounds like a very naive idea. Nonetheless, we believe that it is the only way to mechanically and objectively guide the stochastic search to producing poem-like texts.

  Above we have mentioned a score-relative-to-target strategy for both phonetics and semantics. This seems to be the most concrete method of evaluation, and is what we have chosen to implement in our current system. Certain features, however, most notably those considered to be *preferences* as opposed to *constraints*, do not lend themselves easily towards this strategy. Furthermore, as mentioned above, we would sometimes like the flexibility of allowing the system to operate unguided by such a specific target.

  A naive alternative scoring method is to maintain a tally of points for every occurrence of a feature encountered in a text. This resembles a greedy algorithm search heuristic. For example: applied to the feature of alliteration, if we scored positively for each word that appeared in a line starting with the same phoneme, the final output could become ridiculously riddled with redundant repetitions of rewordings. This might be good for generating tongue-twister-like sentences, but any literary critic would baulk at these results. However, at the moment this is how we implement evaluation of such features, and although we do not intend to go deep into literary theory, we hope to develop a more sophisticated approach.

  For now our aim is to facilitate a modular approach to the evaluation of features, so that each particular type of feature will have its own corresponding "evaluator function". This will allow for more sophisticated approaches and techniques to be easily added in the future.

  Apart from a modular approach, we also aim to parameterize the behaviour of these evaluation functions, e.g. allow a user to set the coefficients and weighting factors that determine the calculation of a certain score. A very interesting prospect is the interfacing of these coefficients with empirical data obtained from statistical literary analysis, or stylometry.

- **Weighting** across features: assuming we have obtained numerical scores for each of the features we are considering, how do we combine them? As in the previous point about parameterizing coefficients of a particular evaluator, we propose to treat the weighting across features in a similar fashion. This parameterization could possibly allow a choice between, say, a preference for rigidly structured poetry and a preference for a more contemporary content-driven poem.

3.2 Evolution

After evaluating a set of candidate solutions and choosing a subset of candidates with the best score, we must then create new variations of them through "mutation". This process can be seen as applying a collection of operators on the chosen candidates. We introduce here three conceptual types of operators, before describing our currently implemented operators in Section 4.6:

- **Add**: "John walked"→"John walked *to the store*"
- **Delete**: "John likes Jill and Mary"→"John likes Jill"
- **Change**: "John walked"→"John *lumbered*"

Due to our integrated architecture, these mutations may occur at different underlying levels of representation of the text. Because these different levels are all interdependent, the operators must take special care to preserve consistency when performing mutation. For example, if the addition of "*to the store*" is viewed mainly as a syntactic addition of a prepositional phrase, the operator would have to update the semantics to reflect this, for instance by adding `destination(w,shop)`. In contrast, if it is viewed primarily as a semantic addition, the operator would have to realize these semantics, one option being the use of a prepositional phrase. Our practice of introducing semantics via a flexible "semantic pool" addresses this issue (see Section 4.3).

Another issue is that these operators can perform non-monotonic modifications on the structures of the candidate solutions, hence our grammar formalism must allow for this.

As it is probably too optimistic to rely on pure random mutation to lead us to a decent poem, we would also like to introduce several heuristic-rich operators. These heuristics would be the encoding of "how to write poetry" guidelines, such as "use 'little' words to 'pad' sentences when trying to fit the metre", and "don't use words with few rhymes at the end of a line". These 'smarter' operators, however, seems to go against stochastic search traditions wherein the operators are deliberately knowledge-poor, relying on the stochastic nature to lead us to the solution. Here, we are adding informedness of heuristics to the whole stochastic process, somewhat analogous to **sampling bias** in stochastic search.

4 Implementation

We are currently in the process of implementing our stochastic search model in Java. In this section we will first briefly discuss our choice of representation for grammar, lexicon and semantics, before describing properties of the architecture and the current implementation of our evaluation and evolution functions.

4.1 Grammar Formalism

Our choice of representation is Lexicalized Tree Adjoining Grammar, or LTAG. For reasons of space, however, we will not explain this formalism in depth. See Joshi and Schabes (1992) for details.

In Tree Adjoining Grammar, a derivation tree is a kind of meta-level tree that records operations performed on elementary trees. In particular, nodes of a derivation tree do **not** signify phrase structure in any way, but rather the process of adjoining and substitution of elementary phrase structure trees. Nodes of a derivation tree are labelled by references to elementary trees, and edges are labelled by the address at which the elementary tree of the child node substitutes or adjoins into the elementary tree of the parent (see Figure 1).

The root node of the derivation tree would introduce the verb, and its two siblings would introduce the subject and object noun phrases. The edges would signify at which NP node the child gets substituted into, effectively informing which is the subject and which is the object.

The common way to deal with TAG trees is by repeatedly performing adjunction and substitution, while having a derivation tree as a record-keeping 'roadmap' of how the tree is derived. However, since we can change and delete portions of our text, the derived tree is problematic since there is no way to "un-adjoin" subtrees. We must always refer back to the derivation tree. In effect, there is no point in maintaining the derived tree throughout the generation process. Instead, the derivation tree becomes our primary data structure, and everything else can be derived from it on demand.

When our operators are said to perform adjunction and / or substitution, they are simply recording the operation in the derivation tree, not actually performing it.

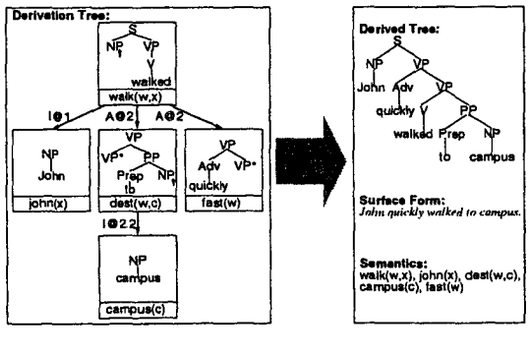


Figure 1: Making the LTAG derivation tree our main data structure

LTAG has the following advantages for our work:

- The *adjunction* operation in LTAG allows flexible incremental generation, such as subsequent insertion of modifiers to further refine the message. This is required as the system builds texts incrementally through an iterative process.
- LTAG provides an extended domain of locality which allows for predicate-argument structures and feature agreements over a structured span of text that need not be contiguous at the surface. This potentially allows for the coupling of poetic features such as rhyming across lines.
- We also adopt an extension to the formalism, Synchronous Tree Adjoining Grammar (STAG), which has been proven useful for paraphrasing purposes Dras (1999).
- LTAG provides an elegant mechanism for our non-monotonicity requirement through the use of the derivation tree. It keeps all syntax and semantics locally integrated at each node, and allows non-monotonic modification of content simply by deleting or replacing the corresponding node.

4.2 Linguistic Resources

At the moment we are still using a very small hand-crafted grammar and lexicon. Like most TAG-based systems, the grammar is a collection of elementary trees, and the lexicon is a collection of words that specify which elementary trees they can anchor. The lexicon also provides phonetic information and lexicon stress, which is extracted from the CMU Pronunciation Dictionary.

A typical lexical entry looks something like this:

```
Orthography:fried
Elementary Tree(s):ITV
Signature:F,Frier,Fried
Semantics:fry(F,Frier,Fried)
Phonetic Spelling:f,r,ay1,d
```

whereas a typical grammar entry looks something like this:

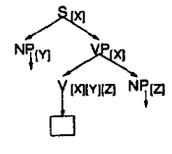


Figure 2: Grammar entry ITV: Intransitive Verb

When binding a lexical entry to an elementary tree, argument structure is preserved by unifying the *signature* of a lexical entry with the signature of its preterminal node. In the above case: (X=F, Y=Frier, Z=Fried).

4.3 Semantics

For our semantics, we follow Stone and Doran (1997) in using an ontologically promiscuous flat-semantics (Hobbs, 1985). The semantics of a given individual is simply the

conjunction of all the semantics introduced by each lexical item, given adjustments for unification of argument structure.

Each individual is associated with a "semantic pool", which is simply a collection of propositions. Unlike the communicative goals of a traditional NLG system, the generator is under no commitment to realize these semantics. The relationship between an individual's semantic pool and its derivation tree's semantics is very flexible. In particular, there is no subsumption relationship either way between them.

Furthermore, in the beginning (the initialization phase) all semantic pools are initialised with a copy of the target semantics. This is ultimately what we hope our resulting poem to "be about". But as time progresses, each individual in a population can evolve and mutate its own semantic pool. Therefore each individual not only differs in form, but also in content.

4.4 Integrated, Incremental Generation

As mentioned above, unity of poetry demands that semantic, syntactic, and lexical information are available at every step of decision making. This calls for an integrated architecture, where there is no explicit decomposition of the process. In our implementation, this is reflected by the fact that individuals are complete structures which maintain all this information, and the semantic, syntactic and lexical operation functions can be applied in any order.

Like most stochastic search algorithms, the process starts with an initialisation phase. Provided with the input of a target semantics and target phonetic form as mentioned in Section 3.1, it then creates a collection of individuals, each corresponding to a minimally complete utterance that *more or less* conveys the target semantics.

What follows is a process of incrementally modifying the utterance to satisfy the target phonetic form while simultaneously attempting to maintain an approximation of the target semantics, as follows:

- During the evaluation phase, a collection of separate evaluators will analyse each individual for its surface form, phonetic information, and semantics, and assign a score (see Section 4.5).
- After every individual has been scored, the set is sorted. The higher ranked individuals spawn "children", copies of themselves which are mutated during the next phase. These children replace the lower ranked individuals, thus maintaining the set size.
- During the evolution phase, a collection of separate operators will be applied randomly on the aforementioned children (see Section 4.6).

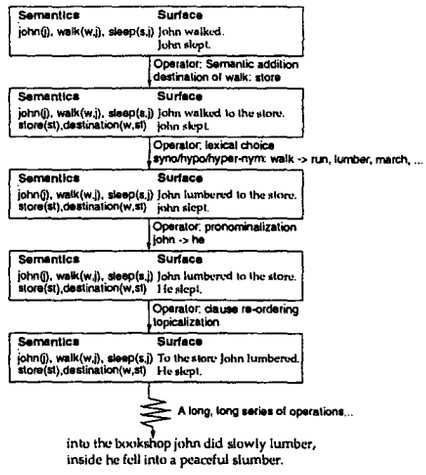


Figure 3: Idealized diagram of a stochastic search

4.5 Evaluators

Unfortunately, at the moment we have only implemented an evaluator for rhythm: the metre evaluator. It works by first dividing the stress pattern of a given utterance into metrical feet of descending/falling rhythm. For instance, the line "There **/once** was a **/man** from Ma**/dras**", has a stress pattern of `(w,s,w,w,s,w,w,s)`. This can be divided into feet as `(w),(s,w,w),(s,w,w),(s)`. In other words, this line consists of a single upbeat (the weak syllable before the first strong syllable), followed by 2 *dactyls* (a classical poetry unit consisting of a strong syllable followed by two weak ones), and ended with a strong beat.

The evaluator compares the metrical configuration of an individual with the target phonetic form by first comparing their number of feet, penalizing those that are either too short or too long. Since each foot, with the exception of the upbeat, contains exactly one strong syllable, this effectively evaluates how close they match in number of strong syllables. It then compares the number of weak syllables between each corresponding foot, once again penalizing the discrepancies, but the penalty coefficient we impose here is less than that of the strong syllables. This provides a natural formalization of the heuristic that strong syllables dominate the effect of a line's metre, and a surplus or missing weak syllable here and there is quite acceptable. For example, (2) sounds more like (1) than (3) does:

(1) The **/cur**few **/tolls** the **/knell** of **/par**ting **/day**
(2) The **/cur**few **/tolls** the **/knell** of ***the*** **/par**ting **/day**
(3) The **/cur**few **/tolls** the **/knell** of ***/long*** **/par**ting **/day**

4.6 Operators

We have currently implemented the following operators:

- Semantic explorer: this operator works with the semantic pool of an individual. Currently it just intro-

duces random propositions into the pool, but with the help of a knowledge-base, it could introduce propositions that are conceptually related to what is already existing in the pool.

- Semantic realizer: This operator is one of the most important ones: it interfaces between the semantic pool and the actual built structure. The semantic realizer will randomly select a proposition from the pool and attempt to realize it by:
  - Selecting all lexical items that can convey the proposition,
  - For each lexical item, selecting all elementary trees that can be anchored by it,
  - For each elementary tree, selecting all nodes in the derivation tree where it can be applied (either adjoined or substituted),
  - Building a list of all these possible nodes and choosing one at random, and inserting the new lexicalized elementary tree at that position.
- Syntactic paraphraser: This operator works by randomly selecting an elementary tree in an individual's derivation tree and trying to apply a suitable paraphrase pair in the manner of Dras (1999). Since all adjunction and substitution information is kept relative to one's parent node in the derivation tree, adjusting for paraphrases (i.e. changing an elementary tree at a certain derivation tree node) is a simple matter of replacing the elementary tree and updating the addressing of the children.

  For example, if paraphrasing a sentence from active to passive form, this would involve exchanging the "Active Transitive Verb" elementary tree at the root to "Passive Transitive Verb", and updating the substitution addresses of the subject and object noun phrases so that the subject now moves to the end of the verb and the object moves to the front.

5 Examples and Discussion

While our system is still in a very early stage of implementation, particularly in terms of evaluators, operators, and linguistic resources, we already have some sample output to present.

For comparison, we first describe our previous attempt at implementing poetry generation, reported in Manurung (1999). This was not a stochastic search model but exhaustively produced all possible paraphrases using chart generation, while simultaneously pruning portions of the search space which were deemed ill-formed from an early stage. It also worked with the target specification of both phonetic form and semantics.

As an example, given the target semantics {cat(c), dead(c), bread(b), gone(b), eat(e,c,b), past(e)} and the target form shown in Section 3.1, but disregarding the rhyme scheme, the chart generator could produce, among others, the following "limerick":

the cat is the cat which is dead;
the bread which is gone is the bread;
the cat which consumed
the bread is the cat
which gobbled the bread which is gone

Since this system does not have the equivalent of a semantic explorer operation (Section 4.6), the output semantics is always subsumed by the target semantics. Moreover, because the chart generator rules out ill-formed subscontituents during the bottom-up construction, the output form always matches exactly the target form. There are no partial solutions or imperfect poems.

In Example 1 below of our stochastic model, there is an appearance of *Luke*, despite not being mentioned in the target semantics. This is due to the semantic explorer operator, which randomly introduces new predicates into the semantic pool. By and large, though, the output does approximate the target semantics. Unfortunately, predicate argument structure and agreement is currently not considered, and this limits the treatment of semantics to a rather trivial account.

The target metre is not precisely followed, but the resulting form is arguably comparable with what a human might produce given the same task.

The resulting score is obtained from the metre evaluator (Section 4.5), and is out of a maximum 1.0.

Example 1:

Input	
Target semantics: {john(1), mary(2), dog(3), bottle(4), love(5,6,7), slow(8), smile(9,10)}	Target form: w,s,w,w,s,w,w,s, w,s,w,w,s,w,w,s
Output (Score: 0.893)	
Surface: *the bottle was loved by Luke* *a bottle was loved by a dog*	Stress: w,s,w,w,s,w,s, w,s,w,w,s,w,w,s

In Example 2, given no semantic input at all, the system will still produce output, but since the semantic explorer operator is still purely random, it resembles word salad. Furthermore, the second sentence is the ungrammatical "ran". This is because this generation effort was terminated prematurely by manual intervention, and at the last iteration before termination the semantic realizer chose to introduce the verb "ran", leaving its subject empty.

Example 2:

Input	
Target semantics: none	Target form: w,s,w,s,w,s,w,s,w,s, w,s,w,s,w,s,w,s,w,s
Output (Score: 0.845)	
Surface: *a warm distinctive season humble mellow smiled refreshingly slowly. ran.*	Stress: w,s,w,s,w,s,w,s,w,s, w,s,w,s,w,w,s,w,s

Although these results can hardly be called poems, nevertheless they succeed in showing how the stochastic hillclimbing search model manages to produce text that satisfies the given constraints, something very difficult for a random word-salad generator to achieve.

6 Related Work

The work reported in this paper is similar in some sense to the work of, among others: NITROGEN (Langkilde and Knight, 1998), a generator that employs a generate-and-test model, using a knowledge poor symbolic generator for producing candidate solutions and ranking them based on corpus-based information for preferences, and SPUD (Stone and Doran, 1996), a TAG-based generator that exploits opportunity arising between syntax and semantics, allowing it to generate collocations and idiomatic constructs.

7 Conclusion

Poetry generation is different from traditional informative generation due to poetry's unity, which essentially means the satisfying of interdependent constraints on semantics, syntax and lexis. Despite our implementation being at a very early stage, the sample output succeeds in showing how the stochastic hillclimbing search model manages to produce text that satisfies these constraints.

References

Kim Binsted, Helen Pain, and Graeme Ritchie. Children's evaluation of computer-generated punning riddles. *Pragmatics and Cognition*, 5(2):309–358, 1997.

Margaret A. Boden. *The Creative Mind: Myths & Mechanisms*. Weidenfeld and Nicolson, London, 1990.

Marjorie Boulton. *The Anatomy of Poetry*. Routledge and Kegan Paul, London, 1982.

Audrey Daly. *Animal Poems*. Ladybird Books, Loughborough, 1984.

Koenraad De Smedt, Helmut Horacek, and Michael Zock. Architectures for natural language generation: Problems and perspectives. In Giovanni Adorni and Michael Zock, editors, *Trends in Natural Language Generation: An Artificial Intelligence Perspective*, number 1036 in Springer Lecture Notes in Artificial Intelligence, pages 17–46. Springer-Verlag, Berlin, 1996.

Mark Dras. *Tree Adjoining Grammar and the Reluctant Paraphrasing of Text*. PhD thesis, Macquarie University, Australia, 1999.

Charles O. Hartman. *Virtual Muse: Experiments in Computer Poetry*. Wesleyan University Press, 1996.

Jerry Hobbs. Ontological promiscuity. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, pages 61–69, Chicago, Illinois, 1985. The Association for Computational Linguistics.

Aravind K. Joshi and Yves Schabes. Tree adjoining grammars and lexicalized grammars. In Maurice Nivat and Andreas Podelski, editors, *Tree Automata and Languages*. Elsevier Science, 1992.

Irene Langkilde and Kevin Knight. The practical value of n-grams in generation. In *Proceedings of the Ninth International Workshop on Natural Language Generation*, Niagara-on-the-Lake, Ontario, 1998.

Samuel R. Levin. *Linguistic Structures in Poetry*. Number 23 in Janua Linguarum. 's-Gravenhage, 1962.

Hisar Maruli Manurung. A chart generator for rhythm patterned text. In *Proceedings of the First International Workshop on Literature in Cognition and Computer*, 1999.

Chris Mellish, Alistair Knott, Jon Oberlander, and Mick O'Donnell. Experiments using stochastic search for text planning. In *Proceedings of the Ninth International Workshop on Natural Language Generation*, Niagara-on-the-Lake, Ontario, 1998.

Nicolas Nicolov. *Approximate Text Generation from Non-Hierarchical Representations in a Declarative Framework*. PhD thesis, Department of Artificial Intelligence, University of Edinburgh, 1998.

Ehud Reiter. Has a consensus on nl generation appeared? and is it psycholinguistically plausible? In *Proceedings of the Seventh International Natural Language Generation Workshop*, 1994.

Mike Sharples. An account of writing as creative design. In Michael Levy and Sarah Ransdell, editors, *The Science of Writing: Theories, Methods, Individual Differences and Applications*. Lawrence Erlbaum, 1996.

Matthew Stone and Christine Doran. Paying heed to collocations. In *Proceedings of the Eighth International Workshop on Natural Language Generation*, pages 91–100, Brighton, 1996.

Matthew Stone and Christine Doran. Sentence planning as description using tree adjoining grammar. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 198–205, Madrid, Spain, 1997. The Association for Computational Linguistics.

Joseph Weizenbaum. Eliza - a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45, 1966.

Towards a Theoretical Framework for Sound Synthesis based on Auditory-Visual Associations

Kostas Giannakis, Matt Smith
School of Computing Science
Middlesex University - Bounds Green
London N11 2NQ - United Kingdom
T: (+44) (0)181 362 6727
F: (+44) (0)181 362 6411
k.giannakis@mdx.ac.uk; m.r.smith@mdx.ac.uk

Abstract

In this paper, we provide a critical review of research efforts that attempt to identify the high-level perceptual dimensions of two distinct sensory percepts, musical timbre and visual texture. These dimensions are tested against a number of evaluation criteria in order to define appropriate sets for further empirical investigation of auditory-visual associations.

1 Introduction

Our age is characterised by a growing interest in the application of computers in arts. Computer technology has become an everyday tool for creative expression in almost every area of human activity. As a result, traditional ways of expression have been transformed and most importantly new forms of electronic art have been introduced. Music, both as a form of art and a field of science, has undergone significant changes due to the application of computers in areas such as sound analysis and synthesis, composition, performance, etc.

Computers can generate sounds either for the imitation of acoustic instruments or the creation of new sounds with novel timbral properties. Almost 50 years of research in computer music laboratories worldwide has resulted in the development of a large body of diverse sound synthesis techniques, e.g. additive synthesis, subtractive synthesis, physical modelling, granular synthesis (see Roads, 1996). Usually, a synthesis technique comprises a set of low-level parameters related to Digital Signal Processing (DSP) modules (*unit generators* in computer music jargon) such as oscillators, filters, etc. A common characteristic of synthesis techniques is that a sound is represented as an object consisting of a large number (hundreds or thousands) of short sub-events that can be controlled by numerous time-varying parameters. Inevitably a vast amount of musical data must be defined and modified (ideally in real time) making the process of creating a sound object a very complex, non-musical and tedious task. Therefore, although it is theoretically possible to create almost any sound using one or more techniques, the main problem with current computer sound synthesis is how the composer interacts with the system in order to create and manipulate sound. This problem becomes even more apparent and complicated when we consider the auditory dimension of timbre and its significance as a compositional tool. Traditional music compositional processes have focused mainly on pitch and duration, and treated timbre as a second-order attribute of sound (Wishart, 1996). However, the development of computer-based sound synthesis tools paved the way for a more tractable exploration of musical timbre. Although current sound design tools provide very sophisticated control of the low-level parameters of sound, there is very little investigation on how a perceptual model of timbre can be incorporated in the design process.

In this paper, we argue that recent developments in the understanding of our auditory mechanisms as well as studies related to our sensory experience can provide a useful theoretical framework for the design of sounds in ways that have a strong cognitive basis. Our research focuses on the mapping between perceptual dimensions of auditory and visual percepts. We propose a metaphor for sound design and representation in computers, which is based on *auditory-visual* associations.

First, we set out to find an appropriate set of perceptual dimensions that can be used for the intuitive control and manipulation of timbre. Second, we argue that a similar set of dimensions can be constructed on the basis of results from studies in the perception of visual texture. Finally, we outline the design of an experiment for the empirical investigation of the cognitive associations between timbre and visual texture.

1.1 A Visual language for Sound Synthesis

The significant role of visual communication in computer applications is indisputable. In the case of music it seems that it is very natural for musicians to translate

non-visual ideas into visual codes (see Walters (1997) for examples of graphic scores from J. Cage, K. Stockhausen, I. Xenakis, and others). In the past, associations between auditory and visual elements (e.g. Isaac Newton's colour-pitch associations) inspired a new artistic movement under the title of *visual music* (e.g. Wells, 1980; Goldberg and Schrack, 1986; Peacock, 1988; Whitney, 1980; Pocock-Williams, 1992). In the domain of sound synthesis, modern systems incorporate graphical editors (e.g. for the drawing of waveforms[1]) and/or on-screen interconnections of graphical objects (e.g. oscillators, filters, etc.). However, the utilisation of a visual language for sound synthesis is still based more on low-level acoustic information (i.e. time-domain and frequency domain representations) and less on how composers actually conceive and externalise compositional ideas that are primarily based on high-level perceptual experiences.

It is useful here to distinguish between two modes of conception: *low-level* and *high-level*. In the low-level mode, a composer conceives and plans compositional actions in terms of DSP instruments using one or more sound synthesis techniques. Subsequently, the composer may use either a low-level (e.g. Csound (Vercoe, 1986)) or a combination of low and high-level interfaces (as in the case of ARTIST (Miranda, 1994) where users first design a DSP instrument and then control it using a natural language interface). In the high-level mode of conception, the internal musical idea may be completely abstract (e.g. a velvety sound). Again, this idea may be externalised by using a low-level interface, a high-level interface, or a combination of both. However, there is no direct high-level interface to match the composer's abstract musical idea. In between musical ideas and sound generated by computers lies the control of a synthesis technique. As a result, the focus of composers has shifted from the high-level musical task of sound design to the low-level and cumbersome process of understanding and controlling the sound production mechanism idiomatic to each synthesis technique. In this study, we argue that a visual language that is based on an investigation of auditory-visual associations can provide a direct high-level interface for the control and manipulation of sound.

As an introduction to auditory-visual associations we present research efforts on colour-sound associations. Previous attempts to model sound using colour (e.g. Padgham, 1986; Caivano, 1994) were based on correspondences that may exist between the physical dimensions of sound and colour. For example, in Caivano's approach, hue is associated with pitch since both these dimensions are closely related to the dominant wavelengths in colour and sound spectra respectively. In the same manner, pure (or high-saturated) colours are associated with pure (or narrow bandwidth) tones whereas low-saturated colours (those that involve wider bandwidths of wavelength) are associated with complex tones and noise. Finally, colour lightness is associated with loudness (black and white represent silence and maximum loudness respectively with the greyscale representing intermediate levels of loudness). In further studies to empirically investigate the validity of these associations, Giannakis and Smith (2000) suggested that pitch and loudness can be predicted by colour lightness and saturation respectively (see Figure 1). This latter study was only concerned with pitch and loudness and involved the use of pure tones in order to neutralise the effect of timbral richness.

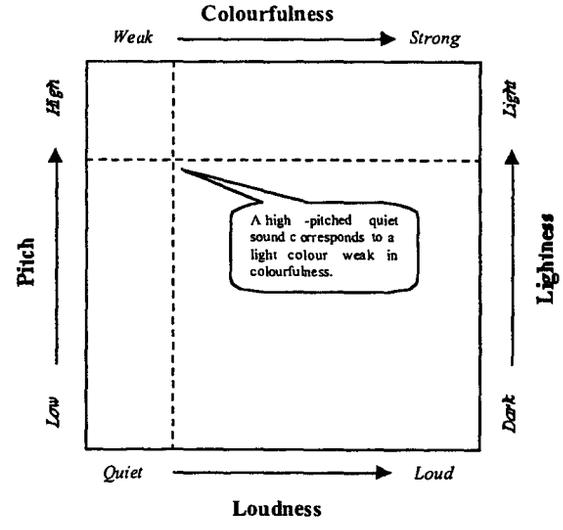


Figure 1: Proposed space for the associations between pitch-lightness and loudness-colourfulness based on Giannakis & Smith (2000).

A natural extension of the above-described studies is the empirical investigation of the associations between dimensions of timbre and dimensions of other visual percepts such as shape, texture, etc. Pitch and loudness are well understood auditory dimensions and both can be ordered on a single scale. In contrast, the perception of timbre is a more complex and multidimensional phenomenon. Recently, visual texture has been proven effective when used in the visualisation of multidimensional data sets (e.g. Ware and Knight, 1992; Healey and Enns, 1998). We have also identified a number of important similarities between timbre and visual texture that suggest further investigation of the potential cognitive associations between these sensory percepts. These similarities will be discussed in more detail in the remaining sections of this paper.

1.2 The Perception of Timbre

Timbre has been defined by the American Standards Association (1960) as "that attribute of auditory sensa-

[1] E.g. the UPIC system by Xenakis (1992)

tion in terms of which a listener can judge that two sounds similarly presented and having the same loudness and pitch are dissimilar". This definition has been strongly criticised (e.g. Bregman, 1990; Slawson, 1985) for being too general and ill-defined. In fact, the term 'timbre' is used in a variety of contexts and it is extremely difficult to agree on a single definition. For example, timbre may refer to a class of musical instruments (e.g. string instruments as opposed to brass instruments), a particular instrument in this class (e.g. violin), a particular type of this instrument (e.g. Stradivarius), the various ways of playing this instrument in order to change the resulting timbre, and so on. Therefore, any attempts to investigate the dimension of timbre should clarify which aspect(s) of timbre are addressed. In this study, timbre is defined as that perceptual attribute which pertains to the steady-state portions of sound. Although temporal characteristics are equally important and necessary for a complete description of timbre (see Grey, 1975), they are more related with the identification of sound sources and their intrinsic behaviour rather than the qualitative characteristics of timbre that are hidden in the steady-state spectrum of sounds.

Many studies attempted to identify the prominent dimensions of timbre (e.g. Bismarck, 1974a,b; Grey, 1975; Plomp, 1976; Ehresman and Wessel, 1978; Slawson, 1985; McAdams, 1999). These studies suggest that there is a limited number of dimensions on which every sound can be given a value, and that if two sounds have similar values on some dimension they are alike on that dimension even though they might be dissimilar on others. However, there is no agreement (with the exception of the dimension of *sharpness*) on the dimensions of timbre that these studies proposed. This is mainly due to the different sets of sounds that were used as stimuli in the experiments (e.g. instrument tones as opposed to synthetic tones) and the different time portions of the sounds that were investigated (e.g. attack transients as opposed to steady states). As a result, these findings hold very well for the limited range of sounds that they refer to but they lack generality of application (see also Bregman, 1990). Nevertheless, these studies suggest that timbre depends on certain characteristics of the sound spectra. Based on the above we can suggest that spectral models for sound synthesis can form the basis for a more intuitive approach to sound design. Spectral models are based on perceptual reality, can be controlled by perceptual parameters, and they provide a general model for all sounds in an analysis/synthesis form (Serra, 1997a). However, research in this area is still young and there is no definite set of appropriate perceptual parameters that can be used effectively in sound synthesis systems.

In the remainder of this section, we first describe dimensions[2] of timbre that have been proposed by the above-described studies and then test them against a number of evaluation criteria. Our goal is to define a set of perceptual dimensions that can be incorporated in further investigations. These criteria can be summarised as follows:

- *Empirical support.* This criterion tests whether a proposed dimension is supported by experimental work.
- *Independence.* This criterion tests whether a perceptual dimension is orthogonal (i.e. independent of changes in other dimensions) or it is somehow correlated with other dimensions (in which case, we can talk of composite dimensions).
- *Measurability.* This criterion tests the existence of concrete measurement methods for perceptual dimensions.
- *Synthesisability.* This is related to the criterion of measurability and refers to existing or potential models of synthesis algorithms that control perceptual dimensions.

1.2.1 Sharpness

Sharpness (other terms include auditory brightness, spectral centroid, etc.) is the most prominent dimension of timbre suggested by the above-described studies. For pure tones, sharpness is determined by the fundamental frequency, i.e. the higher the fundamental frequency, the greater the sharpness. In the case of complex tones, the determining factors for sharpness are the upper limiting frequency and the way energy is distributed over the frequency spectrum, i.e. the higher the frequency location of the spectral envelope centroid, the greater the sharpness (Bismarck, 1974b).

1.2.2 Compactness

Compactness is a measure of a sound on a scale between complex tone and noise, i.e. the difference between discrete and continuous spectra. However, the formulation of such a scale has been proven difficult (e.g. Bismarck, 1974a). Compactness is also related to the concept of periodicity. An ideal periodic (or harmonic) spectrum contains energy only on exact integer multiples of the tone's fundamental frequency. Malloch (1997) suggested that *cepstrum* analysis as a method to measure the periodicity of a sound could also be used in the measurement of compactness.

[2] Dimensions of timbre that refer to temporal characteristics (e.g.) have been excluded from this discussion for the reasons stated earlier.

1.2.3 Spectral Smoothness

Spectral smoothness is a dimension of timbre discussed in McAdams (1999). It describes the shape of the spectral envelope and it is a function of the degree of amplitude difference between adjacent partials in the spectrum of a complex tone. Therefore, large amplitude differences produce jagged envelopes, whereas smaller differences produce smoother envelopes. A formula for the measurement of spectral smoothness can be found in McAdams (1999).

1.2.4 Roughness

Roughness is related to the phenomenon of *beats*. When two pure tones with very small difference in frequency are sounded together, then a distinct beating occurs that gives rise to a sensation of sensory dissonance (Sethares, 1999). In a series of experiments with pairs of pure tones, Plomp (1976) found that roughness reaches its maximal point at approximately 1/4 of the relative critical bandwidth. For complex tones, roughness can be estimated as the sum of all the dissonances between all pairs of partials (see Sethares, 1999).

1.2.5 Discussion

As far as the criterion of empirical support is concerned, the above-described dimensions are the results of rigorous empirical investigations involving musical and/or non-musical subjects within the limitations of their sound stimuli. Usually, two experimental techniques have been employed: multidimensional scaling (e.g. Grey, 1975; Plomp, 1976; Ehresman and Wessel, 1978; McAdams, 1999) and semantic differential scales (e.g. Bismarck, 1974a). The former is based on subjects' judgements of similarity or dissimilarity between sets of stimuli. These similarities and dissimilarities are then represented in the form of a geometric configuration. In semantic differential techniques, subjects are asked to rate sounds along bipolar scales such as sharp-dull, hard-soft, etc. Based on the above, we can argue that all the presented dimensions of timbre satisfy the criterion of empirical support. However, not all dimensions appear to be independent of each other. For example, roughness has been studied as an individual perceptual dimension and has not been part of a larger set of orthogonal dimensions that is usually produced by multidimensional scaling techniques. The dimension of sharpness has been found orthogonal to compactness in studies by Bismarck (1974a) and orthogonal to spectral smoothness in studies by McAdams (1999). Therefore, further investigation is needed to tell whether compactness and spectral smoothness are orthogonal themselves or correlated in some way. These studies have proposed ways of measuring perceptual dimensions of timbre and the reader is referred to the individual studies for more detailed information on computational issues. Finally, these studies have been mainly concerned with the analysis of sounds and there is no explicit discussion about the synthesisability of these dimensions. However, it seems feasible to create synthesis algorithms that are based on the measurement formuli.

1.3 The Perception of Visual Texture

Even though texture is an intuitive concept, an exact definition of texture either as a surface property or as an image property has never been adequately formulated. In this study, texture is considered as a visual percept.

In vision research, there are two main computational approaches to the analysis of texture: the *stochastic* approach and the *structural* approach. The stochastic approach relies primarily on pre-attentive viewing (i.e. when textures are viewed in a quick glance) and is based on statistics and the theory of probability. In the structural approach, a texture is composed of a primitive pattern that is repeated periodically or quasi-periodically over some area. The relative positioning of the primitives in the pattern are determined by placement rules. In a different attempt, Francos et al. (1991) describe a texture model which unifies the stochastic and structural approaches. This model allows the texture to be decomposed into three orthogonal components: a harmonic component, a global directionality component, and a purely non-deterministic component[3].

There are a small number of studies that attempt to identify the perceptual dimensions of visual texture. An early study by Tamura et al. (1978) suggested coarseness, contrast, and directionality as the most prominent perceptual dimensions of texture. The same study also constructed mathematical models for the above dimensions based on extensive psychometric studies. However, the proposed dimensions were based on the authors' subjective views and therefore the question of whether humans use these dimensions in texture judgements was not adequately answered. In another study, Ware and Knight (1992) proposed a visualisation method based on a set of texture dimensions comprising orientation, size, and contrast. Again, the selected dimensions were not empirically derived. In order to address this problem Rao and Lohse (1996) performed a series of experiments that tried to identify the high-level dimensions of texture perception by humans using a variety of experimental designs and statistical methods (e.g. multidimensional scaling, hierarchical clustering, principal components analysis) to analyse and support their results. This latter study confirmed some of the dimensions proposed by earlier

[3] Note the similarity of this approach with a sound synthesis model proposed by Serra (1997b) based on a deterministic plus stochastic model.

studies as being prominent in texture perception. The perceptual space proposed by Rao and Lohse (1996) comprises the following three orthogonal dimensions:

- *Repetitiveness*. This dimension refers to the way primitive elements are placed and repeated over a texture image. The degree of repetition (e.g. periodic, quasi-periodic, random) can be specified and controlled by placement rules.

- *Contrast and Directionality*. This is a composite dimension due to a high correlation coefficient. Contrast is related with the degree of local brightness variations between adjacent pixels in an image (i.e. sharp vs. diffuse edges). The directionality of a texture is a function of the dominant local orientation within each region of texture.

- *Granularity, Coarseness, and Complexity*. These dimensions are very similar to each other (this is supported by high correlation coefficients) and refer to the size (small vs. large) and structure (fine vs. coarse) of the texture grains.

Based on the above discussion and using the same criteria as for the perceptual dimensions of timbre we can suggest that the model proposed by Rao and Lohse (1996) satisfies all the criteria and therefore consists a suitable set of dimensions for the description of visual texture.

1.4 Conclusions and Further Work

In this paper, working towards a theoretical framework of auditory-visual associations, we attempted to combine the findings of various studies in the perception of timbre and visual texture. Timbre and visual texture have been shown to share some very important characteristics. First, timbre and visual texture are both multidimensional perceptual phenomena that can be described by a small set of prominent dimensions. Second, studies in both fields are based on a similar research methodology, i.e. rigorous empirical investigations of how humans perceive and describe sensory percepts. The findings of these studies were evaluated using a number of important criteria. Table 1 summarises the two sets of perceptual dimensions we propose as suitable for further investigation.

Table 1: Identified perceptual dimensions of musical timbre and visual texture.

Timbre	Texture
Sharpness	Repetitiveness
Compactness	Contrast, Directionality
Spectral Smoothness	Granularity, Coarseness, and Complexity
Roughness	

The next step in our research is to design and conduct an experiment based on these sets of dimensions for timbre and visual texture. The objective of this experiment is the identification and investigation of associations (if any) between these auditory-visual dimensions. Based on our discussion, a number of initial hypotheses for such associations can be made. For example, sharpness may be related with contrast, periodicity with repetitiveness, roughness with granularity and coarseness, and finally compactness with complexity. In this experiment, the sound stimuli will consist of steady-state timbres varying systematically in one or more dimensions. Similarly, a collection of textures with varying dimensions will be used for a timbre-texture association task. These textures can be either generated by computer-based texture synthesis algorithms or they can be selected from larger texture databases that are extensively used in texture perception research (e.g. Brodatz textures). The results will provide significant evidence on the analogies between the dimension sets proposed in this paper and will contribute towards the development of a cognitively useful visual language for sound synthesis.

Acknowledgements

Thanks to Prof. Ann Blandford for comments and suggestions on initial drafts of this paper. This research is supported by a Middlesex University research studentship.

References

American Standards Association. *Acoustical Terminology SI, 1-1960*. American Standards Association, 1960

G. von Bismarck. Timbre of Steady Sounds: A Factorial Investigation of its Verbal Attributes. *Acustica*, 30:146-159, 1974a

G. von Bismarck. Sharpness as an Attribute of the Timbre of Steady Sounds. *Acustica*, 30:159-172, 1974b

A. S. Bregman. *Auditory scene analysis: The perceptual organization of sound*. MIT Press, 1990

J. L. Caivano. Colour and Sound: Physical and Psychophysical relations. *Colour Research and Applications*, 19(2):126-132, 1994

D. Ehresman and D. Wessel. Perception of Timbral Analogies. *IRCAM* Reports, 1978

J. Francos, A. Meiri, B. Porat. Modelling of the Texture Structural Components Using 2-D Determi-

nistic Random Fields. *Visual Communications and Image Processing*, Vol. SPIE 1666:554-565, 1991

K. Giannakis, M. Smith. *Imaging Soundscapes.* In press, 2000

T. Goldberg, G. Schrack. Computer-Aided Correlation of Musical and Visual Structures. *Leonardo,* 19(1):11-17. Pergamon Press, 1986

J. M. Grey. *Exploration of Musical Timbre.* PhD Dissertation. Report No. STAN-M-2. CCRMA. Stanford University, 1975

C. Healey, J. Enns. Building Perceptual Textures to Visualize Multidimensional Datasets. In Proceedings *IEEE Visualization 1998*

S. N. Malloch. *Timbre and Technology: An Analytical Partnership.* PhD Dissertation. University of Edinburgh, 1997

S. McAdams. Perspectives on the Contribution of Timbre to Musical Structure. *Computer Music Journal*, 23(3):85-102. MIT, 1999

E. R. Miranda. *An Artificial Intelligence Approach to Sound Design.* PhD Dissertation. University of Edinburgh, 1994

C. Padgham, C. The Scaling of the Timbre of the Piping Organ. *Acustica* 60: 189-204, 1986

K. Peacock. Instruments to Perform Colour-Music: Two Centuries of Technological Experimentation. *Leonardo,* 21(4):397-406. Pergamon Press, 1988

R. Plomp. *Aspects of Tone Sensation.* Academic Press, 1976

L. Pocock-Williams. Toward the Automatic Generation of Visual Music. *Leonardo,* 25(1):445-452. Pergamon Press, 1992

A. R. Rao, G. L. Lohse. Towards a Texture Naming System: Identifying Relevant Dimensions of Texture. *Vision Research,* 36(11): 1649-1669. Pergamon Press, 1996

C. Roads. *The Computer Music Tutorial.* MIT Press, 1996

X. Serra. Current Perspectives in the Digital Synthesis of Musical Sounds. *Formats 1.* Pompeu Fabra University, 1997

X. Serra. Musical Sound Modelling with Sinusoids plus Noise. In C. Roads et al. (eds), *Music Signal Processing.* Swets & Zeitlinger, 1997

W. A. Sethares. *Tuning, Timbre, Spectrum, Scale.* Springer-Verlag, 1999

W. Slawson. *Sound Color.* University of California Press, 1985

H. Tamura, S. Mori, T. Yamawaki. Textural Features Corresponding to Visual Perception. *IEEE Transactions on Systems, Man, and Cybernetics*, 8:460-473, 1978.

B. Vercoe. *Csound.* Computer Music Application. MIT, 1993

J. L. Walters. Sound, Code, Image. *EYE* magazine, 7(26):24-35. Quantum Publishing, 1997

C. Ware, W. Knight. Orderable Dimensions of Visual Texture for Data Display: Orientation, Size, and Contrast. *CHI - ACM Conference on Human Factors in Computing Systems*, 1992

A. Wells. Music and Visual Colour: A proposed correlation. *Leonardo,* 13(1):101-107. Pergamon Press, 1980

J. H. Whitney. *Digital Harmony.* Byte Books, 1980

T. Wishart. *On Sonic Art.* Revised Edition. Harwood Academic Publishers, 1996

I. Xenakis. *Formalized Music.* Revised edition. Pendragon Press, 1992

WASP: Evaluation of Different Strategies for the Automatic Generation of Spanish Verse

Pablo Gervás
Departamento de Inteligencia Artificial
Escuela Superior de Informática
Universidad Europea CEES
28670 Villaviciosa de Odón
Spain
pg2@dinar.esi.uem.es

Abstract

WASP is a forward reasoning rule-based system that takes as input data a set of words and a set of verse patterns and returns a set of verses. Using a generate and test method, guided by a set of construction heuristics obtained from formal literature on Spanish poetry, the system can operate in two modes: either generating an unrestricted set of verses, or generating a poem according to one of three predefined structures (*romance*, *cuarteto*, or *terceto*). Five different construction heuristics are tested over different combinations of two sets of initial data, one obtained from a classic poem and one obtained from a paragraph of a doctoral thesis in linguistics. A set of numerical parameters are extracted from each test, and evaluated in search of significant correlations. The aim is to ascertain the relative importance of size of initial vocabulary, choice of words, choice of verse patterns and construction heuristics with respect to the general acceptability of the resulting verse.

1 Programs that Write Poetry Automatically

The creation of programs that write poetry automatically has been a recurring dream within the AI community, but it has always been assigned a very low priority. Practical applications in the area of natural language processing, such as natural language database interfaces, information retrieval and extraction, automatic translation, and dialogue systems provide more immediate rewards.

On one hand the automatic generation of poetry involves advanced linguistic skills and common sense, two of the major challenges that face AI in general. On the other hand it involves an important amount of creativity and sensibility. These ingredients are very difficult to characterise formally, and very little is known about how they might be treated algorithmically.

On the positive side, poetry has the advantage of not requiring exaggerate precision. If one accepts that the main aim of a poem is to be pleasing rather than conveying a meaningful message, the general problem becomes tractable. The present paper considers how the different parameters that can be controlled by the generating program affect the acceptability of the result. The set of parameters to be monitored are: size of initial vocabulary, choice of words, choice of verse patterns, and construction heuristics. The elusive concept of acceptability of a verse is determined by resorting to hand evaluation by a team of volunteers. By searching for correlations between the strategy and initial data used to generate a verse and the positive or negative evaluation of the resulting verse, information is obtained about the relative relevance of these parameters to the end result.

1.1 Guiding Heuristically the Random Generation of a Verse

Poems written by combining randomly a given set of words rate very poorly with discerning readers. For the words to make sense together, they must be organised according to particular patterns. A possible course of action would be to provide the system with adequately rich lexicon, syntax and semantics for the language involved. Results obtained with inadequate formalisms are too rigid and tend to have a mechanical ring to them. The system presented in this paper resorts to a radical simplification of the underlying linguistic skills. The exhaustive knowledge approach is abandoned in favour of a heuristic engineering solution. Only the barest outline of a grammatical outline is provided (in the form of a verse pattern) to ensure syntactic correctness. Semantic correctness is not enforced, on the understanding that creativity in poetry relies to a certain extent on daring transgressions (such as imaginative metaphors). The aim of the paper is to establish whether acceptable verse may be obtained by controlling other parameters within these initial restrictions. The hope is

to identify whether the elementary ingredients considered can be manipulated smartly enough to produce a pleasing phrase.

1.2 The Effect of the Selection of Initial Data

Under these restrictions, Spanish has been chosen as a test language. The phonetics of Spanish are quite straightforward to obtain from the written word. Most letters in Spanish sound the same wherever they appear in a piece of text, so the metrics, or the syllabic division, of a verse can be worked out algorithmically (1). Spanish scholars have a love for rules, and there is a good set of formal rules (2) describing the conditions that a poem must fulfil in order to be acceptable.

Given such a set of rules, the challenge becomes a simple problem of transforming the given evaluation rules (designed to be applied to an existing poem in order to ascertain its acceptability) into the corresponding construction rules. These rules have to be applied to an initial set of data consisting of:

- a given vocabulary (given a set of words, the poet will choose only some of them, this process of selection must surely play a role in the quality of the final result), and
- a particular choice of ways of combining the chosen words (word order, frequency of adjectives, length of verse...) represented as a set of verse patterns.

The selected vocabulary is a set of words that includes extra information about part of speech roles, number of syllables of each word, position of stressed syllables, and rhyme. The system cannot handle morphological variations, so it considers the singular and plural, masculine and feminine forms of a word as totally distinct (and different tenses of a verb also). This decision reduces the complexity of the generation process to pattern matching between word categories and verse patterns, but it has consequences on the quality of the resulting verses.

The set of valid words is stored as facts of the form:

```
(word (cual luz)
      (numsil 1)
      (acento 1)
      (emp 0)
      (term 0)
      (cat susfem)
      (rima uz))
```

where:

- the field `cual` is the word itself, to be used as key when retrieving the rest of the information
- the field `numsil` shows the number of syllables of the word
- the field `acento` shows the position of the stressed syllable from the beginning of the word
- the field `emp` indicates whether the word starts with a vowel (required to scan a verse)
- the field `term` indicates whether the word ends with a vowel
- the field `rima` contains the rhyme of the word

Each verse pattern is a list of tags, and each tag acts as place keeper for a possible word of the verse. The tag is actually a string that represents information about part of speech, number, and gender of the word that would stand in that particular place in the pattern.

This information is stored as facts of the form:

```
(patron prep artmas adjmas susmas adjmas adjmas)
```

where `patron` is the generic fact name and the following items are tags for the categories of the words in a particular verse.

Patterns act as seed for verses, therefore a pattern determines the number of words in a verse, the particular fragment of sentence that makes up the verse, and the set of words that can be considered as candidates for the verse. By following this heuristic shortcut, WASP is able to generate verses with no knowledge about grammar or meaning.

1.3 Comparing Different Strategies

This paper presents the results of several experiments designed to determine the relative merits of different strategies for generating verses. These strategies are considered with respect to several parameters.

A first group of strategies plays a role in determining simply the number of verses generated from a given set of initial data. These strategies can be tested by generating simple lists of independent verses and evaluating the acceptability of the results. The analysis of the results should determine optimal choices for: (1) the method used to avoid having repeated words in a given verse or poem, and (2) the specific definition used to validate each successive draft of a verse.

A second group of strategies is expected to affect the quality of complete poems generated by the system. These strategies are tested by attempting to generate a number of poems for each strophic form and evaluating the results. Conclusions should provide information about: (1) effect of length of vocabulary, (2) effect of number of patterns, (3) effect of 'informed' assignment of verse borders (4) effect of interaction between patterns and words in the vocabulary (5) effect of extending the words of the vocabulary beyond those present in the initial data

2 A Brief Introduction to Spanish Poetry

This section outlines a few concepts related to Spanish poetry and metric that play a role in the definition of the

WASP system. For more extensive treatment, see (2). A good summary in English is available at (3).

2.1 When and Why a Verse is Valid

Formal analysis of poetry considers the position of stressed syllables over a verse. For the verse to sound pleasing, the prosodic accents must be distributed according to precise patterns. This distribution of prosodic patterns provides the quality of being pleasant to the ear.

For instance, for an eleven syllable long verse to sound pleasing, it needs some of the stressed syllables of its words to fall on certain specific positions. It is not necessary for the stressed syllables of every word in the verse to be in specific positions. It is enough for certain strategic syllabic positions within the verse to have a stressed syllable. The literature (2) requires stressed syllables to fall either on positions 1, 6 and 10; 2, 6 and 10; 3, 6 and 10; 4, 6 and 10; or 4, 8 and 10. In the following examples, stressed syllables falling in key positions are underlined:

en verdes hojas ví que se tornaban	2, 6, 10
nunca fue corazón; si preguntado	3, 6, 10
De tan hermoso fuego consumido	4, 6, 10
soy lo demás, en lo demás soy mudo	4, 8, 10

2.2 Synaloepha: Counting the Number of Syllables

A metric syllable does not always match the corresponding morphological syllable. When a word ends in a vowel and the following word starts with a vowel, the last syllable of the first word and the first syllable of the following word constitute a single syllable. This is known as *synaloepha* (see (2), or (3) for an overview in English), and it is one of the problems that we are facing. For instance, the following verse

bástete amor lo que ha por mí pasado
13 syllables

turns into:

bástete - amor lo **que - ha** por mí pasado
11 syllables

because it shows two instances of synaloepha (marked in bold).

2.3 Building Poems

A *poem* may be an unstructured sequence of verses, but this paper is concerned specifically with poems that make use of known *strophic forms* or *stanzas* (particular patterns of structuring the verses of a poem according to verse length, metre, and rhyme). In such cases, the formal rules that govern the chosen strophic form can be used to guide the generation process.

A poem may consist of a single stanza or several stanzas together (in which case the different stanzas are usually separated by an empty line).

For the present purposes, only three of the simplest strophic forms need be considered:

1. *romances*, a stanza of several verses where all even numbered verses rhyme together and the rhyme of odd verses is free
2. *cuartetos*, a stanza of four verses where the two outer verses rhyme together and the two inner verses rhyme together
3. *tercetos encadenados*, a longer poem made up of stanzas of three verses linked together by their rhyme in a simple chain pattern ABA BCB CDC...

The following are elementary examples of these stanzas:

Type 1: Romance

Por el Val de las Estacas
pasó el Cid a mediodía.
En su caballo Babieca
muy gruesa lanza traía.
Va buscando al moro Abdala
que enojado le tenía.
...

The rhyme of each verse is marked in bold.

Type 2: Cuarteto

Muérome por llamar Juanilla a Ju**ana**,
que son de tierno amor afectos v*ivos*,
y la cruel, con ojos fugit*ivos*,
hace papel de yegua galici**ana**.

The different rhymes of each verse are marked in bold and italic.

Type 3: Tercetos encadenados

Alma a quien todo un dios prisión ha **sido**,
venas que humor a tanto fuego han d*ado*,
medulas que han gloriosamente ard**ido**
su cuerpo dejará, no su cuid*ado*;
serán ceniza, mas tendrán sent**ido**;
polvo serán, mas polvo enamor*ado*.

These three types have been chosen because each shows a different structural characteristic that may affect the overall result. Type 1 presents a recurring rhyme that flows all along the poem. It uses only one rhyme, so many words that rhyme together are required for an acceptable result (our starting data have proven to be poor choices in this respect). Type 2 presents a very simple but rigid structure. It stands for the simplest possible stanza with enough complexity to be distinguishable from prose (the simplification employed with respect to syntax/semantics

makes it difficult for shorter poems to sound acceptable). Type 3 presents a simple structure that recurs throughout the poem, but with the rhyme changing slowly as it moves down.

3 System Description

WASP (Wishful Automatic Spanish Poet) is a forward reasoning rule-based system that takes as input data a set of words and a set of verse patterns and returns a set of verses. Since the aim of the experiment is to compare different methods of generation, WASP is in fact a set of programs, each one applying a different strategy to generate verses. The different programs that integrate WASP are written in CLIPS (4), a rule-based system shell developed by NASA.

WASP operates over a data structure defined as the *draft of the current verse*. This structure is a list of words that is built incrementally. The algorithms used by WASP follow a generate and test pattern. At each stage of the generation process the draft is tested to ensure that the metric conditions are being met. The moment conditions are violated, the draft of the current verse is rejected and the system starts a draft for a new verse.

3.1 Initial Data

The system requires a set of initial data to start the generation process: a vocabulary and a set of patterns. The choice of vocabulary greatly determines the tense and the topic of the poem. The set of verse patterns can be considered as a set of descriptions of past cases, in the sense that it encodes information about important parameters (number of words per verse, rate of adjectives per noun, tense...) while allowing a certain leeway in terms of specific content (particular words) of the present solution.

The set of initial data is obtained as follows. Given a block of text, it is split into fragments of shorter length. All the words in the poem are included in the vocabulary. The resulting fragments of the original text are used to produce the reference patterns. This is done by checking each word of the fragment in the vocabulary and substituting for it the corresponding category.

When an actual poem is used as the original block of text, the existing division into verses can be used. Alternative divisions can be used to test whether this particular decision of the poet plays a significant role in the quality of the results.

In order to compare the effect of the choice of vocabulary and the choice of verse pattern, two distinct set of data are used to test the programs. The first set of data is obtained from a classic Spanish poem, a Sixteenth Century sonnet by Garcilaso de la Vega (5). The second set of data is taken randomly from an academic work in the field of linguistics. A certain paragraph (6) of equivalent size is chosen, all the words in the paragraph are included, and a set of reference patterns is built by splitting the paragraph into chunks of roughly the required size and encoding the necessary information.

3.2 Basic Algorithm

Generation starts with the selection of an appropriate verse pattern, based on criteria designed to ensure that there is a minimum of coherence across verse boundaries. From this pattern an empty draft of the current verse is generated.

The elementary generation cycle can be described as follows:

1. randomly choose from the given vocabulary a word that matches the first category of the current verse pattern
2. append it to the draft of the current verse
3. eliminate the corresponding category from the current verse pattern
4. test whether the resulting verse draft satisfies the conditions of the strategy being used – and the required length of verse in syllables
5. if the conditions are satisfied, iterate from 1.
6. verses that either violate the conditions, or overshoot or fall short of the given number of syllables are rejected

3.3 Strategies for Single Verse Generation

In order to appreciate more clearly the effect on single verse generation, the relevant experiments are carried out without the additional restrictions that determine poem generation. In practical terms, this means that the verse pattern used for each verse is chosen at random from the initial data.

3.3.1 Avoiding Word Repetition

Three different possibilities are considered:

1. simple random combination of the given words into the given patterns
2. ensure no word appears twice in the same poem by eliminating words that have already been used
3. ensure no word appears twice in the same poem by noting down words that have already been used (this allows a rough procedure of garbage collection to avoid losing words used in failed attempts)

Strategy 2 simply deletes used words from the set of available data. If the current draft is subsequently rejected, the words used so far are lost and cannot be used elsewhere.

Strategy 3 makes a note of words that have been used in the current draft and returns them to the set of available data if the draft is rejected.

3.3.2 Validating Successive Drafts of a Verse

Three different possibilities are considered:

1. simply test that the number of syllables is yet smaller than the required length
2. make sure stressed syllables of the words chosen at each step fall into the positions deemed acceptable by the formal rules
3. implement the previous strategy (make sure stressed syllables of the words chosen at each step fall into the positions deemed acceptable by the formal rules) but taking into account the possibility of synaloepha occurring between words

WASP aims for a generic verse length of eleven syllables. For strategies 2 and 3 that apply formal rules for acceptability in terms of position of stressed syllables, we require that the stressed syllable of any word added to a partially completed verse falls either on positions 1, 2, 4, 6, 8 or 10.

3.4 General Strategy for Generating Poems

The generation of poems requires two additional issues to be solved, both related to the restrictions imposed on each verse by the previous verses of the poem. One concerns the choice of verse pattern to use for the next verse. This issue is independent of the particular strophic form sought. The other concerns the rhyme to use for the next verse, and is governed by the particular rules of each strophic form.

3.4.1 Selecting Verse Pattern for the Next Verse

The selection of a verse pattern for the next verse of a poem must take into account the need for coherence between verses across verse boundaries. When operating in the complete poem mode, WASP stores all the verses generated so far, numbered according to their order in the poem. The verse pattern for the next verse is chosen according to the following criteria: the first word category of the selected pattern must occur in some verse pattern of the initial data immediately after the last word category of the previous verse. For instance, if the pattern used for the previous verse was `(patron ... adjmas)`, any verse pattern of the form `(patron susmas...)` is acceptable provided there is a third verse pattern of the form `(patron ... adjmas susmas...)`.

Because verse patterns are produced with no discrimination between patterns that correspond to beginning, end, or middle sections of a sentence, the system must allow the selection of a verse pattern at random if this condition is not met. This solution is an important source of errors in the final results, and it is an obvious candidate for refinement in subsequent versions of the system.

3.4.2 Selecting Rhyme for the Next Verse

The selection of a rhyme for the next verse imposes an additional restriction on the set of verse patterns that are acceptable candidates for continuing a given poem. As well as fulfilling the condition outlined above, the verse pattern for the next verse must end in a word category for which there is a word in the vocabulary with the required rhyme. The simplest method of achieving this is to select the required word and then find a verse pattern that matches the restrictions imposed on its initial word category (by the previous verse) and on its final word (already chosen at this stage).

The system is designed to generate poems using three different strophic forms: *romances*, *cuartetos*, and *tercetos encadenados*.

For all three cases, the first rhymes are fixed by two initial random choices (selection of a verse pattern to start with, and selection of an end word for that particular pattern). This is because there is no initial reference. Once the first verses have been established, WASP ensures that the following verses fit the corresponding stanza (if the starting data and the verse construction strategies will allow it).

In the case of *romances* and *tercetos encadenados* the conditions are formulated using a modulo operation on the verse number, therefore the system has the theoretical potential of generating poems of any length of verses. The actual restrictions faced by the system are imposed by the initial vocabulary (both by the number of rhymes included and by the sheer number of words, since no word repetition is accepted within the same poem).

For *cuartetos*, the system can only generate a single stanza of four verses.

4 Evaluation of Results

Three different sets of experiments were carried out.

In each experiment of the first set, the versions of the system corresponding to different strategies for avoiding word repetition were compared.

The experiments of the second set were designed to evaluate which of the strategies for validating the current draft of a verse gave better results.

For both the first and the second sets of experiments, each competing version of the system attempted to generate a thousand verses, operating in single verse mode. A classic Spanish poem was used to provide initial data, and division of the poem into verses was respected.

The third set of experiments was carried out using only a version of the system that combined the strategies that had obtained better results over the previous sets. Comparisons were established between results obtained for different combinations of initial data. In this set, each competing version attempted to generate twelve poems for each one of the possible strophic forms.

4.1 Avoiding Repetition over Single Verses

Table 1 shows the average percentage results for each strategy. In each case, the program was allowed to carry out 1000 iterations, using a generate and test method. For this part of the evaluation, verse correctness was evaluated automatically using a logic programming application for the analysis of Spanish verse (7). The evaluating application applies strictly the formal rules found in the literature, and validates a verse if it fulfils the required conditions. As such, it constitutes an impartial judge of the correctness of each verse.

The following strategies are compared:

1. simple random combination
2. eliminate used words
3. annotate and replace used words

Table 1: Avoiding Verse Repetition

Version	% Generated	% Correct	% Corr. Gen.
1	35.50	12.90	36.34
2	0.30	0.20	66.67
3	38.40	15.40	40.10

The results for strategy 1 show that, under the minimum of restrictions, only 35 % of the attempts actually generate a verse. This implies that, if strategy 2 is applied, 75 % of the times words are being used up in vain. This matches up with the observed results, where generation drops drastically after the first few attempts. Strategy 3, providing a reasonable solution to word repetition, improves the results of strategy 1 both in terms of number of verses generated altogether and in terms of number of correct verses generated.

For the rest of the experiments, strategy 3 is used.

4.2 Validating Current Draft of a Verse

Table 2 shows results to the second set of experiments. This set was carried out in a similar manner as the above, and evaluated in the same way.

The strategies compared in this case were:

1. simply count the number of syllables
2. count number of syllables and check position of stressed syllables
3. count number of syllables and check position of stressed syllables taking synaloepha into account

Strategy 3 referred to in this table is actually the same as strategy 3 of table 1, and results for it are given again only for ease of reference. The table shows that imposing additional restrictions on the validation of the current draft does not result in a smaller number of verses being generated. In fact, the number of verses generated increases steadily as more restrictions are applied. Furthermore, the increase is noticeable greater when the complete set of restrictions is applied. This can be attributed to the fact that the initial data correspond to a poem that actually fulfils these conditions strictly. The given vocabulary, and the given patterns perform optimally for the complete set of restrictions.

Table 2: Validating Verse Draft

Version	% Generated	% Correct	% Corr. Gen.
3	38.40	15.40	40.10
4	38.90	18.80	48.33
5	46.60	33.60	72.10

For the rest of the experiments, strategy 5 is used.

4.3 Poem Generation and Initial Data

Exhaustive tests were carried for different combinations of the initial data, using generating strategy 5 on all cases.

The following parameters were combined:

- different sources for initial data (poetic or academic text)
- extensions of the vocabulary beyond the original text
- allowing repetitions of words other than nouns, adjectives, adverbs or verbs
- different division of text into patterns
- providing two possible ways of dividing the original text into patterns as part of the initial data
- mixing the vocabulary from one text and the patterns of the other

A total of 504 trials were carried out (14 combinations, and 36 poems, 12 for each strophic form). Many of the resulting poems were either syntactically incorrect, or too short to be considered as poems. For this reason, evaluation took place in two stages.

During the first stage every resulting poem was assigned three numbers: (1) number of verses of the poem, (2) a value for its syntactical correctness, and (3) a value for its esthetical rating. These values are used as a first stage of filtering to avoid wasting evaluation effort on verses that are too short or nonsensical verses unless they have a certain redeeming feature in an esthetical sense. Values were assigned on first inspection by the author.

Syntactical correctness was evaluated using the following scale:

1. the poem is mostly nonsense

2. the poem contains syntactic nonsense

3. the poem can be parsed as a weakly connected fragment of a sentence

4. the poem can be parsed as a strongly connected fragment of a sentence

5. the poem can be parsed as a connected whole

A fragment is considered weakly connected if it can be parsed in some way as a set of independent sentences. A fragment is considered strongly connected if at least some of the verses join together into sentences that make syntactic sense.

Esthetical rating was subjectively evaluated on the following scale [1]:

1. ugly

2. mediocre

3. acceptable

4. pleasing

5. very pretty

Table 3 shows the average results for each combination over the three different types of strophic form.

Poems rating lower than 3, 3, 3 on such a scale were not considered for the second stage of evaluation. This left a total of 45 poems to evaluate.

The second stage of evaluation was carried out by a team of volunteers. Evaluators were given a list of the 45 poems and they were asked to select the best five, and to assign them an order of preference. Each poem was assigned five points if rated first by some evaluator, four if rated second, three if rated third, two if rated fourth and one if rated fifth. The totals were added and the poems were ordered according to the resulting rating.

4.4 Discussion of the Results

The results contain an enormous amount of information, only part of which has been mined at this stage. However, some very interesting conclusions can be drawn from the resulting facts.

Since it had been assumed from the start that if the choice of words and/or the choice of patterns play an important role in determining the quality of a poem, then there should be considerable differences between WASP poems obtained from one or the other set of data (poetic or academic). This hypothesis is validated by the fact that only six of the 45 acceptable poems were generated using the academic set of initial data.

[1] Allowances were made for the fact that verses were the result of a computer program. The scale of 1 to 5 is taken as the bottom end of a 1 to 10 scale for human-generated poems.

Table 3: Validating Verse Draft

Combin.	Rating	Romance	Cuarteto	Terceto
1	Num	4	2	3
	Synt	3	4	3
	Aesthet	2	2	2
2	Num	3	2	2
	Synt	2	3	3
	Aesthet	2	3	2
3	Num	3	2	2
	Synt	3	3	3
	Aesthet	3	3	2
4	Num	1	1	1
	Synt	4	3	4
	Aesthet	3	2	2
5	Num	4	2	3
	Synt	3	2	3
	Aesthet	3	2	3
6	Num	2	2	3
	Synt	4	3	3
	Aesthet	4	3	2
7	Num	4	2	2
	Synt	3	3	3
	Aesthet	3	2	3
8	Num	5	2	3
	Synt	3	3	3
	Aesthet	2	2	2
9	Num	6	2	2
	Synt	2	3	3
	Aesthet	2	2	1
10	Num	3	2	2
	Synt	2	3	2
	Aesthet	1	2	2
11	Num	3	2	2
	Synt	3	3	3
	Aesthet	2	2	2
12	Num	0	0	0
	Synt	1	1	1
	Aesthet	1	1	1
13	Num	0	0	0
	Synt	1	0	0
	Aesthet	1	1	1
14	Num	0	0	0
	Synt	0	0	0
	Aesthet	1	1	1

Overall, only nine of the combinations that were tried managed to produce a poem that went into the final selection. Of these, only one of them was not using an extended version of the original vocabulary. However, that very one did produce the top scoring poem according to the evaluators. This suggests that in general terms the system performs better with a wider choice of vocabulary, unless the random factor in the generation process

actually comes up with a poem that closely mirrors the original one (which is what happened in this case). While it is clear that recovering the original poem is bound to give an acceptable result, this is hardly a desirable solution.

Two of the combinations that produced most top scoring poems were working with vocabularies that had been extended with extra copies of words other than nouns, adjectives, adverbs or verbs (prepositions, articles, pronouns...). These words tend to appear more often than others in poems, and, being usually short, play an important role as cohesive element for longer words that are more difficult to fit into the metric.

Different divisions of the original text into patterns has no great influence on the result once the vocabulary has been extended. This is because an extended vocabulary contains words of different sizes for the same categories. With limited vocabularies, altering the length of verse patterns may result in the desired length not being achievable by combinations of the given words into the shorter patterns.

Mixing data from two different pieces of text (patterns of one, vocabulary of the other) can have drastic negative effects if there is no match between the categories that appear in the verse patterns and the categories represented in the vocabulary. However, it produces very interesting results from an aesthetic point of view. While no way has been found yet to evaluate this fact numerically, it has been observed informally by many of the evaluators and it should be taken into account for further analysis.

4.5 Further Word

The present experiment is intended as preliminary work in a long term project of developing a knowledge based poem generator in Spanish. The results obtained will help to discriminate between the different possible strategies. Additional knowledge and heuristics governing the selection of appropriate verse patterns to follow a given verse might be used either to guide poem construction or to eliminate poor results.

Several interesting insights have been obtained from the analysis of the results presented here. Such cases have been mentioned in the body of the paper wherever appropriate. Better heuristics must be developed for the selection of appropriate pattern for the next verse. Verse patterns should be distinguished in some way according to whether they are beginning, middle or end sections of a sentence. The evaluation procedures are still subject to a great deal of improvement. In a matter where subjective opinion of the reader, special effort must be made to devise an evaluation procedure that provides a rigorous rating without interfering with the natural attitude of the evaluator as reader of a poem.

Acknowledgements

The author wishes to thank to the magnificent team of volunteers that steadily worked their way through reams of verses (particularly obscure ones in many cases) to provide a reasonable evaluation of the performance of the system: Miguel Mulet Parada, Iñigo Eguzguiza, Juan José Escribano Otero, Beatriz San Miguel López, Carlos Bezos Daleske, Alberto Díaz Esteban, Luis Guerra Salas, Carlos Bruquetas, Oscar Rodríguez Polo, María José García García, Javier García Navas, and Celia Rico Pérez.

References

[1] Real Academia Española (Comisión de Gramática), 'Esbozo de una nueva gramática de la lengua española', Espasa-Calpe, Madrid 1986.

[2] Quilis, A., 'Métrica española', Ariel, Barcelona, 1985

[3] Wiliamsen, V.G. and Abraham, J.T., Association for Hispanic Classical Theater web page, ftp://listserv.ccit.arizona.edu/pub/listserv/comedia/poetic1.html

[4] Riley, G., A Tool for Building Expert Systems, http://www.ghgcorp.com/clips/CLIPS.html

[5] T. Navarro Tomás (ed.) Garcilaso de la Vega. Obras, Espasa-Calpe, Madrid, 1973, Soneto XXIII, pp 225.

[6] C. Rico Pérez, 'Aproximación estadístico-algebraica al problema de la resolución de la anáfora en el discurso', tesis doctoral, Departamento de Filología Inglesa, Universidad de Alicante, 1994, chapter 5, pp. 143.

[7] P.Gervás, 'A Logic Programming Application for the Analysis of Spanish Verse', submitted to: First International Conference on Computational Logic, Logic Programming Implementations and Applications stream, Imperial College, London, UK, 24th to 28th July, 2000

NEvAr – The Assessment of an Evolutionary Art Tool

Penousal Machado*; Amílcar Cardoso**
* Instituto Superior de Engenharia de Coimbra, CISUC; Qta. da Nora, 3030 Coimbra, Portugal;
** CISUC, Dep. Eng. Informática, Uni. Coimbra, Polo II; 3030 Coimbra, Portugal;
machado@dei.uc.pt; amilcar@dei.uc.pt

Abstract

The use of Evolutionary Computation approaches to create images has reached a great popularity, leading to the appearance of a new art form – Evolutionary Art – and to the proliferation of Evolutionary Art Tools. In this paper we present and make an assessment of one of these tools: NEvAr. We also systematise and describe the work methodology currently used to generate images. When working with NEvAr we focus on the reuse of useful individual, which we store in an image database. The size of this database, and the importance of its role, led us to the development of automatic seeding procedures, which we also describe.

1 Introduction

In the past few years, a new AI area has begun to emerge, usually named Creative Reasoning. Several aspects contributed to the growth of interest in the study of computational creativity: artificial creative systems are potentially effective in a wide range of artistic, architectural and engineering domains where conventional problem solving is unlikely to produce useful solutions; their study and development may contribute to the overall understanding of the mechanisms behind human creativity; in some ways, the study of creativity can be viewed as the next natural step in AI, considering that we already can build systems capable of solving tasks requiring intelligence, can we build systems that are able to solve tasks that require creativity?

Models of the human creative process (e.g. Dewey (1910), Guilford (1968), Wallas (1926) and De Bono (1986)) may constitute an important source of inspiration to the development of artificial creative systems. Human creativity, however, isn't the only source of inspiration available. When looking at nature, we can see all living species in a permanent struggle for life. Long term survival is connected with the capability of adapting to environmental changes. The survival of the fittest individuals and the recombination of their genetic material is the key element of the adaptation process. The recombination of "good pieces" of different individuals can give rise to new and better ones. Furthermore, the slight modification of individuals' genetic code, can also increase the quality of the individuals.

Over the time, natural selection was capable of producing an incredible amount (and variety) of solutions, *species*, to a common problem, *survival*. Thus, there is no doubt that the evolutionary process is a way of producing innovative solutions (Goldberg, 1998). Whether these solutions can or cannot be considered creative, is a different question, and the answer depends on the way we define creativity. Therefore, and since no uncontroversial globally accepted definition exists, we can consider this to be an open question. However, our current standing is that these solutions can be considered creative.

In the past few years, two Evolutionary Computation (EC) approaches (Genetic Algorithms (GA) and Genetic Programming (GP)) have been used as a mean to implement computational creativity, resulting in the appearance of a set of new applications in areas such as music and image generation, architecture and design.

GA are the most common EC approach in the musical field, some examples are the works of Horowitz (1994), Ralley (1995), Biles (1994), Jacob (1995). However, and in spite of the numerous applications, Wiggins et al (1999), which have studied the performance of this type of systems, defend that these approaches are not ideal for the simulation of human musical thought. In the field of image generation, GP is the most used approach. Examples of works in this field are: Dawkins (1987), Sims (1991), Todd (1993), Rooke (1996), which resort to GP to evolve images, and Baker (1993), where GP is used to evolve human faces. GP has also been successfully applied in the fields of design (Bentley, 1999; Graf, 1996)

and animation (Sims, 1991; Angeline, 1996; Ventrella, 1999).

Due to the difficulty of creating an evaluation function in domains such as image or music generation, most of the above mentioned systems use Interactive Evolution (IE). In IE systems the user evaluates the individuals, thus guiding evolution. In the musical field we can already find several systems that resort to automatic evaluation (e.g. Horner et al (1991), McIntyre (1994), Spector (1994, 1995), Hodgson (1996,1990,1999), Papadopoulos et al (1998)). In image generation, the picture is quite different: as far as we know there has been only one attempt to automate fitness assignment, the work of Baluja et al (1994). However, the results produced by this system, which uses neural networks to evaluate images, were disappointing.

The core subject of this paper is the assessment of NEvAr as a tool. In Section 2 we make a brief overview of the previous work in Evolutionary Art Tools, focusing on the most prominent systems. In Section 3, we introduce NEvAr and describe the used evolutionary model. Section 4 concerns the assessment of NEvAr and the description of the work methodology currently employed. This methodology gives emphasis to the reutilization of good individuals, which are stored in a database. The difficulties of managing an increasingly large database led to the study of seeding procedures, which will be described in Section 5. Finally, in the 6th section we make some overall remarks and draw some conclusions.

2 State of the Art

In the past few years, the use of IE to the generation of images has achieved a great popularity. The source of inspiration of most of these applications can be found in Richard Dawkins book "The Blind Watchmaker", in which the author suggests the use of a GA to evolve the morphology of virtual organisms, *biomorphs*. In these systems, the evolution is guided by the user accordingly to hers/his aesthetic criteria. This inspired the works of K. Sims (91) and W. Latham et al (92), which can be considered as the first applications of IE in the field of the visual arts, and are usually considered as the most influential works in this area. The success of these approaches has led to the emergence of a new art form, "Evolutionary Art" (EA), and also to the proliferation of IE applications in this field, usually called Evolutionary Art Tools.

In spite of the increasing number of this type of applications, few are the ones that can be compared favourably with the above mentioned works. The vast majority of these applications adds nothing new to these works, and are, frequently, inferior both in terms of potential and results. Moreover, few are the ones that have been thoroughly tested, i.e. in most cases there was no attempt to use them to create art. Therefore, it seems safe to say that the classification of these applications as Evolutionary Art Tools is misguiding, and that few are the applications that deserve this name. In this restricted set, we can include the works of: K. Sims (91), W. Latham and S. Todd (92), S. Rooke (96), Vetrella (99). The description of the characteristics of these systems and the analysis of their potential is clearly beyond the scope of this paper. These systems share many features, most notably: they resort to GP, use IE and have been successful in the generation of visual artworks.

3 NEvAr

NEvAr (Neuro Evolutionary Art) is an evolutionary art tool, inspired in the works of K. Sims (1991) and R. Dawkins (1987). It allows the evolution of populations of images from an initial one, and resorts to IE. In this section, we will make a brief description of the evolutionary model used in NEvAr. NEvAr is in many ways similar to the application developed by K. Sims (91), namely in what concerns the representation of the individuals and the used genetic operators. Therefore we won't make a description of these aspects.

For the current purpose, it is enough to say that in NEvAr the individuals are represented by trees. The genotype of an individual is a symbolic expression, which is constructed from a lexicon of functions and terminals. In NEvAr, we use a function set composed mainly by simple functions such as arithmetic, trigonometric and logic operations. The interpretation of a genotype results on a phenotype, i.e. an image. All the genetic manipulations (e.g. crossover, mutation) are performed at the genotype level. In Figure 1, we present two images generated with NEvAr and in Figure 2 some images generated by the mutation and crossover of the genetic code of these images.

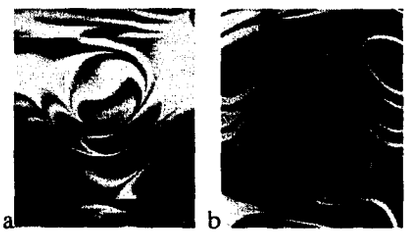


Figure: 1 –Two images created with NEvAr.

Figure 2: The top row images were created through the mutation of image *a* of Figure 1. The bottom row images result from the crossover of images *a* and *b* of Figure 1.

As the images of Figure 2 show, the mutation and crossover operations can produce interesting and unexpected results. The high plasticity, which is inherent to the used representation and, to some extent, to GP approaches, allows the generation of radically different, yet fit, phenotypes from similar genotypes.

3.1 The Model

In NEvAr, the assignment of fitness is made by the user and, as such, she/he has a key role. The interaction of human and computer poses several problems (e.g. limited population size, limited runs). The fact that NEvAr is an interactive tool has the advantage that a skilled user can guide the evolutionary process in an extremely efficient way. She/he can predict which images are compatible, detect when the evolutionary process is stuck in a local optimum, etc. In other words, the user can change its evaluation criteria according to the context in which the evaluation is taking place.

In the design of NEvAr's model, we took under consideration these idiosyncrasies. In Figure 3, we show the model of NEvAr. From here on, we will call *experiment* to the set of all populations, from the initial to the last, of a particular GP run. NEvAr implements a parallel evolutionary algorithm, in the sense that we can have several different and independent *experiments* running at the same time. It is also asynchronous, which means that we can have an *experiment* that is in population 0 and another one that is in population 100. Additionally, we can transfer individuals between *experiments* (migration).

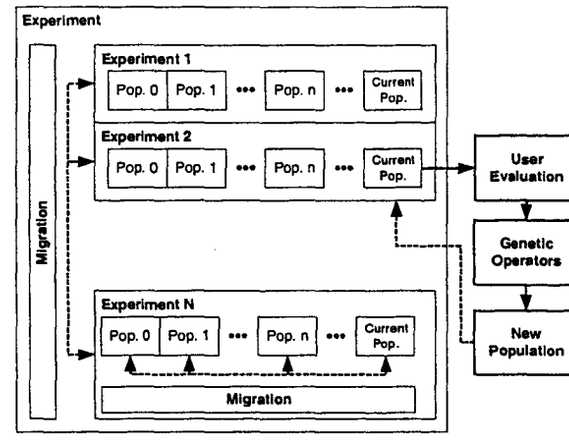


Figure 3: The evolutionary model of NEvAr. The active *experiment* is depicted in grey.

We will illustrate the use of this model through an example. Suppose that the user creates two different *experiments*, *a* and *b*, the initial population of *a* is randomly generated and has size N, and the initial population of *b* has size 0. The user focuses his efforts in *experiment a* and evaluates the individuals of successive populations generated by NEvAr. When the user finds images that she/he likes, she/he adds these images to the current population (in this case the population 0) of *experiment b*. If at a given point the user feels that the evolutionary process would benefit if the next population was generated by the combination of the individuals of the current population with individuals previously transferred to population *b*, she/he adds those individuals to the current population and the evolutionary process continues.

At a certain point, the user decides to focus on *experiment b*, and orders the generation of a new population from the current one (population 0), which is composed, exclusively, by individuals transferred from *a*. Thus, the initial population of *experiment b* is not random, but exclusively composed by fit individuals that were originally generated in other *experiments*. In fact, *experiment b* can be seen as a database of images, which may be used to initialise future *experiments*. We may generalise this approach by organising a gallery of images.

NEvAr also allows the migration within *experiments*. This feature is important due to the limited size of each population, since it allows the revival of images from previous populations. It is also possible to go back to a previous population and change the evaluation of the individuals, which allows the exploration of different evolutionary paths.

4 Working with NEvAr – The artistic point of view

NEvAr is an Evolutionary Art Tool, therefore the main goal is the production of artworks. Its analysis must be performed with this in mind. Like any other tool, NEvAr requires a learning period. To explore all the potential of a tool, the user must know it in detail and develop or learn an appropriate work methodology. The results, and user satisfaction, depend not only on the tool but also on its mastering.

Additionally, the evaluation of the results (images) can only be made by the user that generated them. The evaluation of an art tool can only be made by the artist using it. The key aspect is that the artist must review her/himself in the produced artworks. Thus, the fact that a tool can generate "interesting" images is irrelevant from the artistic point of view. What is really important is that the produced artworks convey the artistic ideas of the artist. In other words the artist must be able to express her/himself through the use of the tool.

The images generated with NEvAr during the early stages of experimentation were clearly disappointing. This failure didn't result from the lack of power of the tool, but from our lack of expertise in its use. Next we will present the work methodology that we currently use to generate images with NEvAr.

4.1 The Process

The creation of an artwork encompasses several stages, such as: genesis of the idea, elaboration of sketches, exploration of the idea, refinement, and artwork execution. The methodology that we propose can be considered, in some way, analogous. It is composed by four main stages: Discovery, Exploration, Selection and Refinement.

These stages can be described, concisely, as follows: the stage of Discovery consists on finding a promising evolutionary path, which, typically, corresponds to evolving a promising set of images from an initial random population (genesis of the idea); in the second stage, Exploration, the "ideas" evolved on the previous stage are used to generate images of high aesthetic value (exploration of the ideas); the Selection stage involves choosing the best produced images; the selected images, when necessary, will be subjected to a process of Refinement, whose goal is the alteration of small details or the correction of imperfections (final execution of the artwork).

Our empirical experience allows us to classify the Discovery stage as the most crucial of the process, and, together with the Exploration stage, the one in which the faculties of the user are more important.

Discovery corresponds to the genesis of the idea, therefore it is inappropriate to approach this stage with pre-conceived ideas regarding the final aspect of the artwork. In other words, it is impossible in practice (yet tempting) to think on an image and use NEvAr to evolve it. This is probably the most important aspect to retain, because it contrasts with what is usually expected in a tool, i.e. that it allow the implementation of an idea. This aspect can be viewed as a weakness, but it is also the distinguishing feature and strength of NEvAr (and other evolutionary art tools). A conventional art tool only plays an important role in the artistic process in stages subsequent to the generation of the idea. Furthermore, the idea frequently determines which tool will be used in its execution, since some are more adequate than others. NEvAr, however, plays a key role in the generation of the idea. Its influence is noticeable through all the artistic process and in its main creative stage. In NEvAr, the artist is no longer responsible for the creation of the idea, she/he is responsible for the recognition of promising concepts. More precisely, the idea results from an evolutionary process, and is created by the artist and the tool, in a (hopefully) symbiotic interaction.

In the Exploration stage the initial idea is already set and we are dealing with images of high aesthetic value. Through the recombination of these images, we explore a space of forms which is smaller than the one explored in the discovery stage, and is therefore more thoroughly searched. The Exploration stage can prolong itself conducting the artist to a point which, at least apparently, has nothing to do with the original one. Like in the Discovery stage, the expertise of the user is determinant to the success of this stage. With the accumulation of experience, the user learns how to distinguish between promising paths and ones that lead nowhere, to predict which combinations of images produce best results, how to manipulate crossover and mutation rates in order to produce best results, etc.

The Selection stage can be divided in two different ones, one that is concurrent with the evolutionary process, and one that is posterior. During the stage of Exploration, the best images (according to the user criteria) are added to a different *experiment*, that works as a gallery. As stated

before, NEvAr stores all populations, which allows the review of the evolutionary process and the addition to the gallery of images that were previously neglected. This revision is highly recommended, and a substantial amount of time should separate the generation of the images and its review in order to allow the necessary distance between generation and criticism.

The Refinement process usually occurs separately from the *experiment* that generated the image. The common procedure is to initialise a new *experiment* with the image that we want to refine (i.e. the initial population of this *experiment* will be composed by the image and, in some cases, similar ones). The generation of new populations, from this initial one, allows the exploration of a search space in the vicinity of the image that we want to refine. In Figure 4 we present some images created with NEvAr.

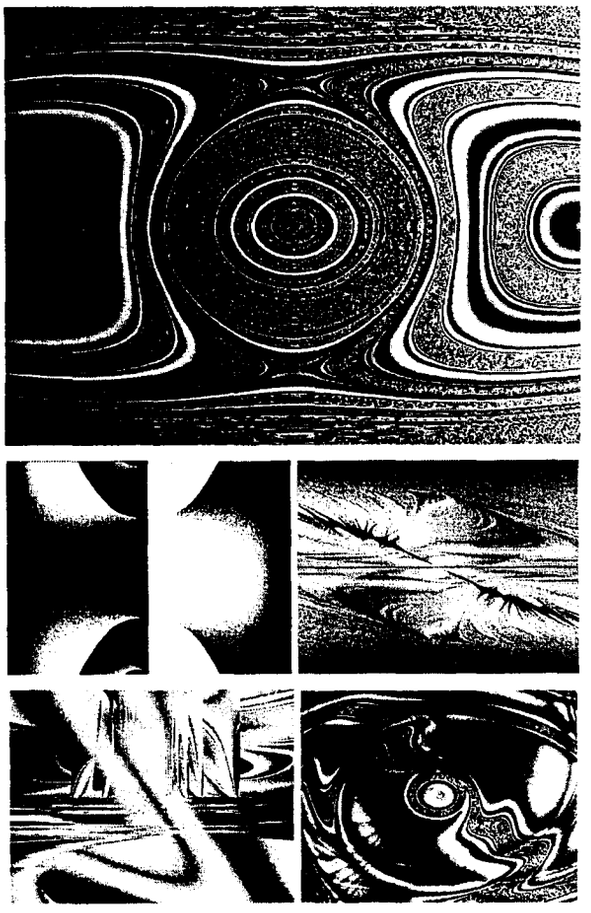


Figure 4: Some examples of images created with NEvAr. Additional images can be found in the CD-ROM accompanying P. Bentley (1999).

4.2 Image Database

One of the weaknesses of EC approaches lies on the fact that they do not have long-term memory (although we can view multiploidy as a limited memory mechanism). The use of several *experiments* allows the accumulation of individuals, which can be used in later experiences. Thus, we can create a Knowledge Base (KB) of images.

The KB has been used mainly in two situations: To initialise new *experiments* and to add individuals to the current population of an *experiment*. The goal of the first form of use is to shorten, or even avoid, the initial stages of the evolutionary process (Discovery and Exploration). The addition of previously generated individuals to the current population usually follows an opportunistic reasoning. There are several situations in which this may be useful, for instance, to avoid a local optimum, or when we find an image whose combination with a previously created one is previewed as promising.

The KB is playing an increasingly important role in the process of image generation, and is currently a priceless feature of the system. The size of the KB is also increasing rapidly and, consequently, the KB is becoming harder to manage and use. This led us to the development of automatic seeding procedures, which we will describe in the following section.

5 Seeding

The development of automatic seeding procedures is part of our ongoing research. In this section we describe our current approaches, which should be considered preliminary.

Our idea is inspired in Case Based Reasoning, and can be described as follows: the user chooses an image, and the seeding procedure selects, from the database, similar ones, to initialise the GP *experiment*. To implement this idea, we need to develop a similarity metric, i.e. a way to compare images. Unfortunately, this task is not trivial.

In our first attempt we used the root mean square error (*rmse*) among two images, which is usually applied to evaluate the error involved in image compression, as similarity metric. Minimum error implied maximal similarity. The similarity between two images, a and b, was given by the following formula:

$$rmse\ sim_{a,b} = \frac{100}{1+\sqrt{rmse_{a,b}}} \qquad (1)$$

The experimental results showed the inappropriateness of this approach. This failure can be easily explained, the goal is to find images that are similar to the eye and not "mathematically" similar images. To illustrate the shortcomings of *rmse* based similarity, we resort to an example: consider two images composed by alternate vertical black and white stripes of one pixel width, one starting with a black stripe and the other with a white one; these images will be almost indistinguishable to the eye, however, the *rmse* among them will be maximal.

Our current idea is to compare images according to their properties. It is a well known fact that image complexity affects the performance of image compression methods, i.e. it is easier to represent compactly a simple image than a complex one. Moreover, some compression methods work better with some types of images than with others.

JPG compression was designed for the representation of natural images. In this type of images, colour transition is usually smooth. Although adequate for these images, the performance of *JPG* severely degrades when dealing with images possessing abrupt colour transitions (e.g. a black and white text image). Fractal Image Compression takes advantage of the self-similarities present on the images and will, therefore, perform better when these similarities are high.

Our previous experience with image compression methods led us to believe that we could use the quality of the compression to develop a similarity metric. For the scope of this paper, we will define compression quality as:

$$\frac{compression\ ratio}{rmse}, \tag{2}$$

and compression complexity as the inverse.

We use two different compression methods: *jpg* and fractal based. The fractal image compression algorithm makes a quad-tree partitioning of the image. By changing the maximum depth of the tree, we can specify, indirectly, the limits for the error involved in the compression. During compression, the colour information is discarded, the images are converted to greyscale and then compressed.

Let's define *IC* as the compression complexity resulting from the use of the *JPG* method; *PC* as the compression complexity resulting from the application of the fractal based approach. We use two different maximum tree depths, N and N-1, therefore we have PC_1 and PC_2.

To compare two images, *a* and *b*, we start by calculating *IC*, PC_1 and PC_2, for each of them. The similarity between images *a* and *b* is given by the following formula:

$$sim_{a,b} = \frac{1}{1+\sqrt{|IC_a - IC_b| + |PC1_a - PC1_b| + |PC2_a - PC2_b|}} \tag{3}$$

In Figure 5, we present a subset the images belonging to the database. In Table 1, we present the *IC*, PC_1 and PC_2, measures for each of them as well as the similarity of these individuals with images 9 and 14 of the population.

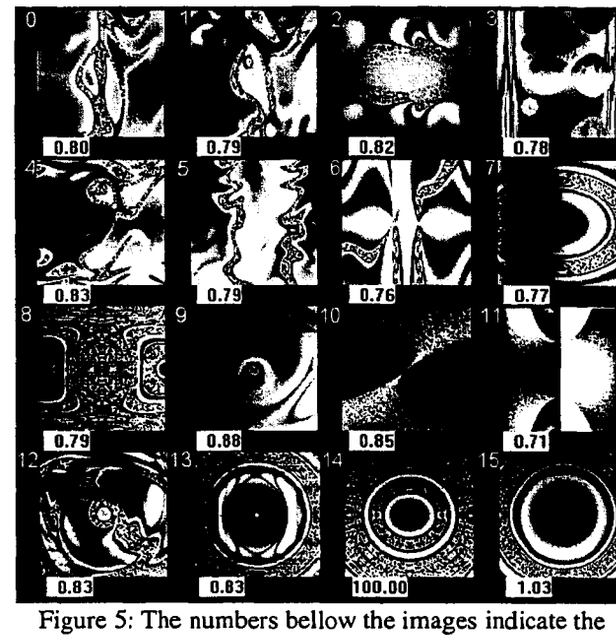


Figure 5: The numbers bellow the images indicate the *rmse* similarity to image 14 (see formula 1). According to this metric, the closest image is image 15, which is good, and the second closest is image 9, which is bad.

Ordering the individuals according to their similarity to image 14, yields the following list: {14, 8, 13, 12, 15, 4, 5, 7, 0, 11, 10, 6, 3, 1, 2, 9}. This ordering seems to be appropriate, the major deficiency being that individual 7 is considered less similar than individuals 4 and 5. The comparison to image 9 gives the ordered list: {9, 2, 1, 3, 6, 10, 11, 0, 7, 5, 4, 15, 12, 13, 14, 8}, which also appears to be approximately correct. Image 9 is characterised by its fluid and organic forms, and so are individuals 2, 3, 6, 0, and, although in a lesser degree, individuals 10 and 11.

Table 1: The IC, PC_1 and PC_2 measures for each of the images presented in Figure 5, and the similarity among these images and individuals 14 and 9 of the same figure.

Image	CI	CP1	CP2	Similarity to 14	Similarity to 9
0	5.053	19.228	5.957	10.397	17.500
1	4.455	10.503	4.646	9.790	22.703
2	2.926	5.518	2.403	9.365	37.261
3	4.085	11.256	5.957	9.879	21.529
4	6.401	21.357	7.057	10.697	16.189
5	5.965	21.663	6.504	10.650	16.365
6	4.694	13.395	4.988	9.976	20.486
7	5.744	19.373	6.795	10.503	16.981
8	12.125	91.074	25.331	16.948	8.349
9	2.399	3.413	2.200	9.239	100.000
10	4.593	12.839	5.883	9.989	20.359
11	5.113	14.434	6.244	10.129	19.170
12	8.736	42.895	13.636	13.765	11.673
13	7.978	45.669	13.523	14.062	11.506
14	11.518	71.164	21.835	100.000	9.239
15	6.891	34.791	10.861	12.181	13.032

The initialisation of a new population is based on the similarity to an image chosen by the user. The seeding procedure uses the similarity values as fitness, and roulette wheel selection for choosing which images will become part of the initial population. We do not allow the repetition of images.

Although the experimental results are still preliminary, the seeding procedure based on compression quality seems to produce good results, indicating that the comparison of images based on their characteristics, namely on their complexity, is appropriate. This, of course, suggests that taking into consideration other types of features of the images (e.g. edges, colour, outline, etc.) may also prove useful.

It is also important to notice that the compression methods described can be applied to any image. Therefore, we can compare the database images with ones that where not generated with NEvAr. We still haven't explored this possibility, nevertheless we believe it can produce interesting results.

6 Conclusions and Further Work

From the artistic point of view, we consider NEvAr to be a tool with great potential. Through the use of NEvAr, the artist is no longer responsible for the generation of the idea, which results from an evolutionary process and from the interaction of artist and tool. Thus, the use of NEvAr implies a change to the artistic and creative process. It is important to notice that, in spite of these changes, the artworks obey the aesthetic and artistic principles of the artist. The use of NEvAr implies an abdication of control; however, this lack of control isn't necessarily negative. The artist can express her/himself through the use of the tool and review her/himself in the works created.

One of the erroneous conceptions about evolutionary art tools is that the generation capabilities of a system are deeply connected with the used primitives. Our experience with NEvAr shows that this is wrong. What is necessary is a set of "basic" primitives that can be combined in a powerful way.

The inclusion of a long term memory mechanism is extremely important, since it allows the reuse of previously generated ideas. It is also the basis for the inclusion of Case Based Reasoning mechanisms in NEvAr. Preliminary experiments indicate that the inclusion of these mechanisms can create interesting results and further extend the capabilities of our system.

Interactive evolution proved to be a very powerful technique. This can be explained by the fact that the user can use other criteria besides fitness to evaluate the individuals, and thus guide the evolutionary algorithm more efficiently. We are currently studying ways to automate fitness assignment. Our initial idea was to train a neural network and use it to automate this task. We currently feel that full automation is not attainable on short term. Our current idea is to use neural networks (as well as other techniques) as a filter that eliminates individuals that are clearly undesirable.

Acknowledgements

This work was partially funded by the Portuguese Ministry of Science and Technology, under Program PRAXIS XXI.

References

P. J. Angeline. Evolving Fractal Movies. *Genetic Programming: Proceedings of the First Annual Conference*, MIT Press, Stanford University, CA, USA, pp. 503-511, 1996.

E. Baker. Evolving Line Drawings, *Technical Report TR-21-93*, Harvard University Center for Research in Computing Technology, 1993.

S. Baluja, D. Pomerlau, and J. Todd. Towards Automated Artificial Evolution for Computer-Generated Images. *Connection Science*, 6, 325-354, 1994.

P. Bentley, (Ed.) *Evolutionary Design by Computers*, Morgan Kaufman, 1999.

J. Biles A genetic algorithm for generating jazz solos. *International Computer Music Conference*, Aarhus, Denmark, 1994.

R. Dawkins, R. *The Blind Watchmaker*, W.W. Norton & Company, Inc., New York, 1987.

E. De Bono. *El pensamiento lateral. Manual de la creatividad.* Barcelona, Spain: Paidós (in Spanish), 1986.

J. Dewey. *How we think.* Boston: D. C. Heath, 1919.

D. Goldberg. The design of innovation: lessons from genetic algorithms, lessons for the real world, *IlliGAL Report N° 98004, Department of General Engineering*, University of Illinois at Urbana-Champain, 1998.

J. Graf, and W. Banzhaf. Interactive Evolution for Simulated Natural Evolution. *Artificial Evolution*, Vol. 1063, Springer Verlag, pp. 259-272, 1996.

J. P. Guilford. *The Nature of Human Intelligence.* McGraw-Hill, New York, 1967.

P. Hodgson. Understanding Computing, Cognition, and Creativity. *University of the West of England*, 1990.

P. Hodgson. *Gene-e Evolutionary Dance Music Program*, 1996.

P. Hodgson. Understanding Computing, Cognition, and Creativity. *AISB Symposium on Musical Creativity*, Edinburgh, UK, 1999.

A. Horner, D. Goldberg. Genetic algorithms and computer-assisted composition. Genetic Algorithms and Their Applications: Proceedings of the *Fourth International Conference on Genetic Algorithms*, pp. 427-441, 1991.

D. Horowitz. Generating Rythms with Genetic Algorithms. *International Computer Music Conference*, Aarhus, Denmark, 1994.

B. L. Jacob. Composing with Genetic Algorithms. *International Computer Music Conference*, 1995.

W. Latham, and S. Todd. *Evolutionary Art and Compuers*, Academic Press, Winchester, UK, 1992.

R. A. McIntyre. Bach in a Box: The Evolution of Four-Part Barroque Harmony Using Genetic Algorithms. In *IEEE Conference on Evolutionary Computation*, 1994.

G. Papadopoulos, and G. Wiggins. AI Methods for algorithmic composition: A Survey, a Critical View and Future Prospects. *AISB Symposium on Musical Creativity*, Edinburgh, UK, 1999.

D. Ralley. Genetic algorithm as a tool for melodic development. *International Computer Music Conference*, 1995.

S. Rooke. The Evolutionary Art of Steven Rooke, *http://www.azstarnet.com/~srooke/*, 1996.

K. Sims. Artificial Evolution for Computer Graphics. *ACM Computer Graphics*, 25, 319--328, 1991.

L. Spector, and A. Alpern. Criticism, culture, and the automatic generation of artworks. *Proceedings of Twelfth National Conference on Artificial Intelligence*, AAAI Press/MIT Press, Seattle, Washington, USA, pp. 3--8, 1994.

L. Spector, and A. Alpern. Induction and Recapitulation of Deep Musical Structure. *International Joint Conference on Artificial Intelligence*, IJCAI'95 Workshop on Music and AI, Montreal, Canada, 1995.

J. Ventrella. Animated Artificial Life. *Virtual Worlds - Synthetic Universes, Digital Life, and Complexity*, Heudin, J. C. (ed.) Perseus Books, pp. 67-94, 1999.

G. Wallas. *The art of thought.* Nova Iorque: Harcourt Brace, 1926.

G. Wiggins et al. Evolutionary Methods for Musical Composition. *International Journal of Computing Anticipatory Systems*, 1999.