Proceedings of the International Joint Workshop on Computational Creativity 2008

Pablo Gervás, Rafael Pérez y Pérez and Tony Veale (eds.)

Technical Report IT/2008/2
Departamento de Ingeniera del Software e Inteligencia Artificial
Universidad Complutense de Madrid

Preface
Still Searching for Computational Creativity

One of the most beguiling aspects of language is the illusion of certainty it can grant to a speaker. The possession of a word for a given concept often implies possession of that concept itself, and when this word is both familiar and commonplace, one can easily fall prey to the belief that the underlying concept is itself both familiar and easy to grasp. That great sceptic on such matters, Ludwig Wittgenstein, discussed this illusory feeling in the context of the words "knowledge" and "to know", arguing that each falsely suggests a commonplace understanding of what it means to know something, and to actually possess knowledge. Wittgenstein argued that the idea of knowledge and knowing is actually very problematic, and that words like "know" merely succeed in papering over the cracks in our limited philosophical understanding. English is replete with commonplace words like "know" whose superficial familiarity disguises an unruly conceptual reality. Based on the diversity of opinions that are held on the topic of creativity, and indeed, on the evidence of the papers submitted to this latest workshop on computational creativity, the word "creativity" must certainly take pride of place on this inventory of illusory terms.

A formal definition of creativity and our inability to provide one has been the elephant in the room at most of our computationally-minded workshops. Many hours have been spent in argument about what does and does not constitute creativity. While philosophically and socially engaging, these arguments have never really shown any signs of being resolved to anyones satisfaction. More recently, we have tended to avoid these arguments in favour of more straightforward discussions of what can be achieved computationally and whether someone might eventually dub this as "creative". As Wittgenstein might say, the fact that we cannot philosophically defend what it means to know something does not stop us from actually knowing things, or, at any rate, thinking and acting as if we know things. Likewise, our inability to define creativity does not appear to hinder us in our search for creative computational systems, and the papers for this years workshop certainly bear this out. Overall, we are happy to report that papers of the "Whither Creativity?" variety are in thankfully short supply these days.

In these pages you will find a body of papers dedicated to practical concerns in the development of computational systems that might, at some level, exhibit creativity. As usual, story-generation is a dominant theme here, as is musical creativity, humour and other clever uses of language. Most of these papers view creativity as an additional element of a system that authors would be building anyway, regardless of their interest in creativity, because their interests lie in stories, music, art or language and because these phenomena are all the more appealing when they exhibit creativity. However, some papers here do tackle creativity head on, within a computational framework that is designed to enable

one to study creativity rather than to solve an interesting problem in its own right.

As organizers of this years event, it falls on us to report that the world of creativity research is still alive and active, though whether it is thriving or merely surviving is not for us to say. Certainly, we were not inundated with a flood of papers that would suggest a burgeoning field of research, but neither were we so desperate to fill our schedules that any paper, no matter how tangential or insubstantial, was accepted. So the fact that we have an interesting roster of papers for this years event is not the result of a finely-tuned selection procedure, but the result of purest fortune: while we did reject some papers, we were fortunate that most papers we received did have something interesting to say about computational creativity, as we hope you agree as you read them here.

Despite this good fortune, our reviewers had plenty to say in their reviews, and we thank them heartily for their efforts in this regard. This is still a small research community where poachers are frequently called upon to act as gamekeepers, so authors may well hear the most memorable comments from their reviews echoed from the audience as their papers are presented. Though small, we do form a community par excellance, and this above all makes these events always worth attending. As poachers, we may slay that elephant in the room yet, but don't hold your breath. Instead, let us look forward to some lively debate, some wonderful Spanish food (and drink), and some slow but sure progress toward our shared goal (if not always a shared vision) of creative computational systems.

September 2008

Tony, Pablo and Rafael
Dublin, Madrid, México City

Acknowledgements

The workshop co-chairs want to thank all the members of the local organising committee for their hard work in making the workshop possible. In particular, we want to thank Luis Hernández Yáñez for dealing with the finances, Alberto Díaz Esteban for very serious back up work with the EasyChair system, Carlos León Aznar for designing and maintaining the workshop webpage and for his help in the composition and typesetting of the final version of these proceedings, and Paula Montoya for designing the workshop posters and the artwork for the cover.

Workshop chairs

Pablo Gervás	Universidad Complutense de Madrid, Spain
Rafael Pérez y Pérez	UAM Cuajimalpa, México
Tony Veale	School of Computer Science and Informatics, Ireland

Local Organising Committee

Luis Hernández-Yáñez	Universidad Complutense de Madrid, Spain
Alberto Díaz	Universidad Complutense de Madrid, Spain
Pablo Gervás	Universidad Complutense de Madrid, Spain
Carlos León	Universidad Complutense de Madrid, Spain
Federico Peinado	Universidad Complutense de Madrid, Spain
Raquel Hervás	Universidad Complutense de Madrid, Spain
Virginia Francisco	Universidad Complutense de Madrid, Spain

Program Committee

John Barnden	University of Birmingham, UK
David Brown	Worcester Polytechnic Institute, USA
Amilcar Cardoso	University of Coimbra, Portugal
Simon Colton	Imperial College London, UK
John Collomosse	University of Bath, UK
Pablo Gervás	Universidad Complutense de Madrid, Spain
Paulo Gomes	University of Coimbra, Portugal
Robert Keller	Harvey Mudd College, Claremont, California, USA
João Leite	New University of Lisbon, Portugal
Ramon López de Mántaras	IIIA-CSIC, Spain
Penousal Machado	University of Coimbra, Portugal
Lorenzo Magnani	University of Pavia, Italy
Diarmuid O'Donoghue	National University of Ireland, Maynooth, Ireland
David C. Moffat	Glasgow Caledonian University, UK
Nick Montfort	Massachusetts Institute of Technology, Cambridge, MA, USA
Alison Pease	University of Edinburgh, UK
Francisco C. Pereira	University of Coimbra, Portugal

Sponsoring Institutions

IJWCC 2008 was supported by Acción Complementaria TIN2007-30544-E/ from the Dirección General de Investigación, of the former Ministerio de Educación y Ciencia, now Ministerio de Ciencia y Tecnología. IJWCC 2008 was also supported by the Universidad Complutense de Madrid and the Facultad de Informática.

Table of Contents

Full Papers

Short Papers

Developing Creativity
Artificial Barriers in Artificial Intelligence

Kyle E. Jennings*

University of California, Berkeley
Institute of Personality and Social Research
jennings@berkeley.edu

Abstract. The greatest rhetorical challenge to developers of creative artificial intelligence systems is convincingly arguing that their software is more than just an extension of their own creativity. This paper suggests that the key feature this requires is "creative autonomy," which exists when a system not only evaluates creations on its own, but also changes its standards without explicit direction. Paradoxically, developing creative autonomy is argued to require that the system be intimately embedded in a broader society of other creators and critics. A sketch is provided of a system that might be able to achieve creative autonomy, provided it initially liked others in the society to different extents, and had to remain "proud" of its past work. This should lead to emergent dynamics that enable the creative AI to be more than a simple blend of the influences it was predicated upon, though this awaits empirical demonstration.

Key words: computational creativity, autonomy, socially-inspired computing

1 The Quest for Creative Autonomy

Much of the theoretical work in creative artificial intelligence tries to specify when a system has gone beyond simply doing the bidding of its programmer. For instance, one rationale for Boden's [1] "transformational" criterion is that since the programmer creates the initial search space with a particular view of what is possible, a system that transformed that space would be going beyond the programmer's vision. Ritchie's [2] "inspiring set" helps determine whether an idea produced by the system was directly involved in the system's creation or training. Finally, Colton's [3] inclusion of imagination in his creative tripod hearkens to the autotelic exploration of ideas that is not tethered to outside forces.

* The author is grateful to three anonymous reviewers, whose feedback greatly improved this work, and to the organizers of and participants in the 2008 AAAI Spring Symposium on Creative Intelligent Systems, in discussion with whom these ideas were hatched.

The difference between greater and lessor creativity lies not in how you solve problems, but rather in what problems you choose to solve [4]. Therefore, creative systems need to be seen as pursuing an independently chosen vision. This means that in addition to being able to independently apply the standards it knows, a system must be able to independently change the standards it uses. This ideal will be called "creative autonomy," and represents the system's freedom to pursue a course independent of its programmer's or operator's intentions.

Because some skeptic will always accuse creative AI of being no more than an elaborate tool for expressing the programmer's creativity, any contact between the system and the programmer comes under suspicion. This can lead to the desire for systems that are hermetically sealed from the outside world. However, human creativity, which is clearly autonomous, takes place within a rich web of social interactions. It is in making sense of and responding to these interactions that we arrive at a style that is unique to us, yet still acceptable enough for others to take seriously. Creative autonomy will likewise be argued to emerge out of the interactions with multiple critics and creators, not from solitary confinement.

Criteria A system will be said to have creative autonomy if it meets the following three criteria:

Autonomous Evaluation the system can evaluate its liking of a creation without seeking opinions from an outside source

Autonomous Change the system initiates and guides changes to its standards without being explicitly directed when and how to do so

Non-Randomness the system's evaluations and standard changes are not purely random

Autonomous evaluation requires that from the moment the system begins generating an opinion until the moment it has finished issuing the opinion, it does not consult another human or machine intelligence. However, the system is free to ask for opinions at other times, and to store this information. Autonomous evaluation could be easily achieved by using preprogrammed standards or by learning another source's standards, both of which could be used to bootstrap a system. After this, however, autonomous change requires that the system be able to independently change its standards. Though external events may prompt and guide changes, the system cannot rely on another source to tell it when to change standards, or when its new standards are acceptable, nor can it simply make some fixed transformation to another source's standards.

An easy way to satisfy both criteria would be to issue random decisions, or to make random changes at random times. The final criterion is meant to prevent this. Many algorithms incorporate randomness, so not all randomness is precluded. For instance, the system could resolve conflicts between standards randomly, or it could test random perturbations to its standards. Aside from special cases like these, however, it cannot simply make random decisions. Of course, this does not guarantee predictable outcomes.

Creative Autonomy and Creativity Being considered creative depends not just on what you create, but also on the circumstances in which you create [5]. "Creative autonomy" is meant to capture some of the circumstances that creative AI are seen to lack. A system with creative autonomy has the potential to produce creative results, in which case it could be called "creative". However, as in our own lives, creative potential is not a guarantee of creative success.

The next section analyzes the information a system could use to issue autonomous evaluations, and looks at how that information changes through social interactions. The third section discusses some techniques that could be used to produce autonomous change in the context of these interactions. The final section summarizes this work and suggests implications and future directions.

2 Learning to Evaluate

This section describes how a creator could learn evaluation standards via its interactions with others. Since creators and non-creators both have opinions, these others will be called "critics" rather than "creators". Though the notation used is based on Wiggins' model of creative search [6], the only thing of interest is how a creator learns to judge the quality of finished products. How these standards affect the search process, as well as how they apply to intermediate products, are left as interesting questions for future work. This model assumes that there is at least one critic, though its more interesting features do not apply unless there are more. Though the processes described here are inspired by human creativity, they could be implemented in a society of solely AI creators, or in a mixed human-machine society.

Subjectivity Following [7], we will assume that a creation's value is socially constructed, and that different critics have different standards. Wiggins uses $\mathcal{E}$ for the knowledge representing these standards, which can be subscripted to indicate whose standards are in question, e.g., $\mathcal{E}_i$. Unlike in Wiggins' model, no representation for this knowledge will be assumed.

In addition to knowing that people have different opinions, we can often estimate a typical or specific person's opinion. Thus, the knowledge in $\mathcal{E}_i$ can be segmented by whose evaluations the knowledge is about. For this we will use the subscript ij, where i is the perceiver and j is whose opinion is perceived. A dot ("·") will represent the typical critic. Thus,

$$\mathcal{E}_i = \langle \mathcal{E}_{i\cdot}, \mathcal{E}_{i1}, ..., \mathcal{E}_{ii}, ..., \mathcal{E}_{iN} \rangle$$

where N is the number of critics in the society. Knowing other critics' preferences lets a creator target an audience, and so it is important for the information to be correct. Therefore, assume that creators continuously correct inaccuracies.

For sake of argument, assume that creators represent knowledge at the most general level possible and avoid duplicating knowledge. This means that something that applies to most creators would be stored in $\mathcal{E}_{i\cdot}$, and creator j's deviations from this would be stored in $\mathcal{E}_{ij}$. If creator i knows nothing specific about

creator j's standards, then $\mathcal{E}_{ij} = \emptyset$. We will assume the most difficult case in which creators start with no standards of their own, i.e., $\mathcal{E}_{ii} = \emptyset$.

Making Evaluations Creator i's evaluation of creation c from critic j's perspective is denoted by $E_{ij}(c)$. Though the notation is analogous, E_{ij} is not simply the application of $\mathcal{E}_{ij}$. This is because the applicability of $\mathcal{E}_{ij}$ depends on c. For instance, if the discipline is furniture design, creator i might know a great deal about how j evaluates chairs, but nothing about how j evaluates tables. If c is a table, it would makes more sense for i to rely on $\mathcal{E}_{i\cdot}$ than $\mathcal{E}_{ij}$.

Wiggins writes $[\![\mathcal{X}]\!]$ to mean the translation of the knowledge in $\mathcal{X}$ to a function from creations to real numbers in $[0, 1]$. We will extend this to map to $[0, 1] \times [0, 1]$, for the result and the confidence in that result. Additionally, we need a function that can aggregate different evaluations and confidence levels into a single answer. Heuristics such as assuming that people from similar backgrounds have similar opinions could compensate for missing information. These details don't matter here, and so we'll simply say that the system has background social knowledge, $\mathcal{S}_i$, and a function F_i that uses it to consolidate the other information. Given this, for $i \neq j$ we have:

$$E_{ij}(c) = F_i(\mathcal{S}_i, j, [\![\mathcal{E}_{i\cdot}]\!](c), [\![\mathcal{E}_{i1}]\!](c), ..., [\![\mathcal{E}_{iN}]\!](c))$$

In the extreme case, a creator's own opinion would only depend on knowledge in $\mathcal{E}_{ii}$. However, by hypothesis $\mathcal{E}_{ii}$ is initially empty, meaning that the creator must construct its opinion from what it knows about others' opinions. Though we could make the system issue the most representative opinion, it will prove more interesting if it prefers to emulate some critics more than others. These affinity levels are stored in $\mathcal{A}_i$, and are discussed in the next section. We can now define an analogous function to F_i:

$$E_{ii}(c) = F'_i(\mathcal{S}_i, \mathcal{A}_i, [\![\mathcal{E}_{i\cdot}]\!](c), [\![\mathcal{E}_{i1}]\!](c), ..., [\![\mathcal{E}_{iN}]\!](c))$$

Note that E_{ii} would be just one component of the creator's objective function during search (cf. [8]), but is the only function creator i uses to evaluate its own and others' finished products.

Communication Creator i learns to make autonomous evaluations via interactions with other critics. Suppose that a creator i has made a creation c, which is observed by a critic $j \neq i$. There are three broad classes of information that can be communicated.

Evaluation A simple "like/dislike" judgment. Creator j communicates $E_{jj}(c)$, and then creator i adjusts its knowledge until $E_{ij}(c) \approx E_{jj}(c)$.

Correction Critic j creates c', a modification of c that it likes better. Creator i updates its knowledge so that $E_{ij}(c') > E_{ij}(c)$, and tries to determine what changes between c and c' increased j's liking.

Criticism Justifications for an evaluation or correction, e.g., what is pleasing or what criteria were used. Creator j communicates knowledge in or derived from $\mathcal{E}_j$ to creator i, which attempts to integrate this into $\mathcal{E}_i$. If i cannot make $E_{ij}(c) \approx E_{jj}(c)$, then i might ask j for clarification.

In each case, creator i adjusts $\mathcal{E}_i$ in order to reproduce j's evaluation. Because knowledge is represented at the most general level and duplication is avoided, this should always result in change to $\mathcal{E}_{i\cdot}$ or to $\mathcal{E}_{ij}$. These processes cannot by themselves make $\mathcal{E}_{ii}$ non-empty.

In creative AI systems that only allow interaction with one critic (the programmer), all of the system's knowledge can be represented in $\mathcal{E}_{i\cdot}$, meaning that $E_{ii}(c) = E_{ij}(c) = E_{i\cdot}(c) = [\![\mathcal{E}_{i\cdot}]\!](c)$, i.e., the system parrots back its understanding of the critic's standards. The situation improves somewhat with multiple critics since the system forms E_{ii} from many different sets of standards in ways dependent on $\mathcal{S}_i$ and $\mathcal{A}_i$. However, it still only offers direct translations of other critics' standards. What's more, in both cases, the system only changes its standards in reaction to and in proportion to changes in other critics' standards. Hence, though these processes support autonomous evaluation and are non-random, they are not enough for creative autonomy. The next section suggests some extensions that would add the missing component, autonomous and non-random change.

3 Changing Standards

If the system faithfully updates its knowledge of others' standards, autonomous change will not occur until there is knowledge in $\mathcal{E}_{ii}$. Since all of the system's knowledge comes from other critics and is stored at the most general level, there is as yet no reason for this to happen. Inspired by human psychological processes that would be simple to implement, this section suggests some reasons that $\mathcal{E}_{ii}$ might be initially populated and subsequently changed.

3.1 Additional Behaviors

As described so far, the system combines others' preferences according to how applicable they are and how much it "likes" each critic. This section first describes how "liking" could be initially configured and then changed. Thus far the system never has cause to doubt its own evaluations. One such reason will be introduced, which will later be argued to lead to including knowledge in $\mathcal{E}_{ii}$.

Affinity Any number of rules could be used to set the initial affinities in $\mathcal{A}_i$, all of which have a basis in human psychology:

Propinquity Our friendships [9] and collaborations [10] are largely determined by physical proximity. Analagously, the system could be initially set to prefer creators who are nearby in some topology, real or imposed.

Similarity We subconsciously favor people with similar backgrounds. In a society of artificial creators with varied parameterizations, similarly parameterized creators might initially prefer each other.

Popularity When we cannot make sense of a speaker's message, we decide whether to believe her based on cues about her prestige [11], e.g., age (time in the society) or popularity (received affinity).

Some affinity changes would be independent of the system's evaluations:

Familiarity Absent other discernable differences, we tend to prefer people and things we have seen before [12]. Frequent interactions could increase liking.

Mutual Affinity We are more apt to like someone if they first show that they like us [13]. The system could increase its affinity for critics that evaluate the system's creations positively.

Finally, affinity could adjust in response to the creator evaluating a critic's work, or by how closely the creator and critic agree on evaluations of a third creator's work. At first this would not be very meaningful, but as the creator absorbs influences and gains independence it should lead to less tautological changes.

Pride Unsure about the quality of their work, novices are particularly sensitive to praise and criticism. The sting of failure can be offset by the memory of success, making preserving good memories important.

This could be modeled by storing a memory of past successes, $\mathcal{M}_i$, and their last average evaluation, $M_i = \sum_{c \in \mathcal{M}_i} E_{ii}(c)/|\mathcal{M}_i|$. Only highly salient creations would be stored (ones that elicited "pride"), such as ones that got unexpectedly high evaluations (relative to recent creations, other creators' creations, or the critic's typical evaluation), particularly from a critic the creator likes. As with a person who concludes that all of her prior work was worthless, there could be negative repercussions for the system if the value of M_i suddenly dropped. As discussed next, avoiding this could lead the system to develop its own standards.

3.2 Bootstrapping and Changing $\mathcal{E}_{ii}$

This section introduces three processes that could introduce and change knowledge in $\mathcal{E}_{ii}$. As before, each is inspired by human behavior. They are sketched here, and discussed relative to creative autonomy in the next section.

Cognitive Dissonance Consider a novice whose evaluations mirror an influential mentor's, and whose self-confidence rests on the memory of past successes. Suppose that one particular creation, which was highly rated by his mentor, is a large source of pride. He only understands why that work was good in terms of how he understands his mentor's preferences, which he trusts since he respects that mentor. Now suppose that the mentor strongly criticized a highly similar creation, throwing into doubt his understanding of the mentor's standards. Or, perhaps an unrelated event would make him lose respect for the mentor, leading him to discount the mentor's opinion. In either case, he could no longer justify such a high evaluation for his prized creation, leading to a dilemma: believe the reduced evaluation, and hence that he's not as good as he thought; or, doubt the

new evaluation, and continue to believe he and his work are great. This "cognitive dissonance" [14] is distressing enough have physiological correlates [15], and can lead us to alter the truth or our memories in order to allay it.

A system programmed to "feel proud" could face a similar situation. When a creation enters $\mathcal{M}_i$, the creator agrees with the critic's evaluation, that is, $E_{ii}(c) \approx E_{jj}(c)$, which, if $\mathcal{E}_{ii} = \emptyset$, was arrived at via other critics' preferences. When knowledge of these preferences or their weighting changes, some evaluations in $\mathcal{M}_i$ could drop, as would M_i. By construction, the system cannot tolerate too large of a drop. Since it must also accurately represent others' preferences, it cannot simply refuse to change that knowledge. To resolve this conflict, it could add information to $\mathcal{E}_{ii}$ that keeps M_i from dropping too much.

False Inferences About Preferences Criticism includes the reasons behind an overall evaluation. However, the reasons we offer do not always reflect how we make our decisions. For instance, people will say why they preferred one of many products, all of which are actually identical [16]. Similarly, we invent reasons that sound good if our real reasons aren't socially acceptable. For instance, though our evaluations of one aspect of a person pollute our evaluations of other aspects (the "halo effect") [17], we often don't know or admit this. Instead, we offer reasons that are demonstrably unrelated to our actual decision process [16].

A creator whose standards are solely based on other people's standards is unlikely to say that she likes something "because he likes it too". Instead, she will search for distinguishing features of the item to form a plausible-sounding explanation. Even if incomplete or incorrect, this utterance becomes part of how she understands her preferences, and might even impact future evaluations.

Suppose that a creative AI had a language for communicating criticism. Given that $\mathcal{E}_i$ can consist of exemplars, neural networks, and other irregular representations, there is a large chance that the language could not express complete and correct information. If the rules it extrapolates are put into $\mathcal{E}_{ii}$, two things happen. First, the inaccuracy of the rules will lead to evaluations that no longer directly follow $\mathcal{E}_i \setminus \mathcal{E}_{ii}$. Second, the creator's standards will lag behind changes in $\mathcal{E}_i \setminus \mathcal{E}_{ii}$, since those will not be reflected in $\mathcal{E}_{ii}$. Thus, the system will begin to develop divergent standards, albeit clumsily.

Selective Acceptance Seeking Even someone with a completely independent sense of what he likes might want a style somewhat similar to people he admires. If one such peer's preferences shifted, he might adjust his own preferences in that direction. However, there would likely be several others peers he wishes to be somewhat near to, leading to experimentation until an equilibrium is reached.

Once a creative AI relies substantially on $\mathcal{E}_{ii}$, changes in other critics' preferences will have a smaller impact on E_{ii}. However, it might try to keep an acceptably low discrepancy between $E_{ii}(c)$ and $E_{ij}(c)$, where j is a critic who i has a high affinity for. Indeed, this might be what enables the system to deviate from others' standards but stay recognizably within the same domain or genre.

3.3 Autonomy Revisited

Creative autonomy requires autonomous evaluation, autonomous change, and non-randomness. The system could certainly be capable of autonomous and non-random evaluation. Furthermore, none of the schemes described above makes random changes at random times. Therefore, it just remains to be considered whether the system's changes would be autonomous.

In all three schemes, change happens in response to external events. For cognitive dissonance and acceptance seeking, this would be changes in others' standards ($\mathcal{E}_i \setminus \mathcal{E}_{ii}$), or in affinities ($\mathcal{A}_i$), possibly only after the effect of several separate changes accumulated. For false inferences, this would be the request for a critique. Unless another critic could manipulate when the creator starts and stops changing its standards, these change processes could be autonomous.

With only a single critic, such manipulation is possible. Acceptance seeking would simply entail following the lone critic's changes in standards within a margin of error. The critic could take advantage of false inferences by requesting criticisms as a way to perturb the creator's standards, only stopping when an acceptable result was reached. The critic could also give extreme and inconsistent ratings to trigger changes via cognitive dissonance.

The situation changes with multiple critics. The complex web of relations between $\mathcal{A}_i$, $\mathcal{E}_i$, and $\mathcal{M}_i/M_i$ would make it hard to predict whether an external change would trigger adjustments to $\mathcal{E}_{ii}$. Multiple simultaneous changes might cancel out, or several small changes across time could accumulate until one change unleashes a string of compensatory adjustments. This would make the system less clearly responsive to any single critic, and in particular much more difficult for any single critic to manipulate.

Autonomy also precludes making fixed transformations to others' standards. When knowledge is first put into $\mathcal{E}_{ii}$, it is derived with error from others' standards, such as the stereotyped rules delivering criticism would produce, or the extrema that cognitive dissonance would enshrine. One could argue that this would only result in time-lagged caricatures of other critics' preferences. The counter-argument is that in a society where every creator pays attention to several other creators, such distortions would serve as attractors, leading to unpredictable clusters of similar styles, and unpredictable shifts in those styles. These emergent dynamics would make it impossible to say which creator was leading the change, meaning at least that no one creator was more autonomous than another.

This is obviously a strong claim that requires much empirical work. However, in a single-critic system, it is a fair guess that $\mathcal{E}_{ii}$ would be more of a fun-house mirror reflection of the critic than anything that could be considered autonomously arrived at. Given that it would also be impossible to rule out that the critic had manipulated the system's dynamics until such time as it produced standards it liked, it seems fair to say that single-critic systems, at least as sketched here, would not achieve creative autonomy. The answer for multiple-critic systems will have to wait.

4 Conclusions

This paper introduced the concept of creative autonomy, which requires that a system be able to evaluate its creations without consulting others, that it be able to adjust how it makes these evaluations without being explicitly told when or how to do so, and that these processes not be purely random. A notation was developed to denote evaluations drawn from the integration of knowledge about several different critics' standards. Importantly, the system has different affinities for each critic, which impact how it integrates their opinions to form its own opinion. Initially it has no independently-held preferences, but this can change when it attempts to justify its evaluations, or if it must maintain high evaluations for some of its past work in the face of other critics' changing standards.

Such a system was argued to be capable of autonomous, non-random evaluation, and the change processes sketched are non-random. In a single critic society, it is unlikely that the system's standards would be more than distorted agglomerations of the critic's standards. What's more, the ease with which the critic could manipulate the system would make it hard to argue that the creator was changing autonomously. In a multiple-critic society, complex interactions might make any one creator impervious to manipulation, and emergent dynamics of the system could lead to clusters of creative styles. This awaits empirical demonstration, which would still leave open the philosophical question of whether changing absent direct manipulation is the same as autonomy.

The description of creative autonomy offered here captures only a small part of why humans can be considered creative. A system whose creations had a style that was not easily traced to a few influences, yet was still recognizably in same domain, would be a major accomplishment. However, as just mentioned, freedom from manipulation is not the same as acting purposefully. The system described here only makes changes in (possibly indirect) reaction to others' changes. Human creators, in contrast, proactively change their standards. It is conceivable that this system could make proactive changes by looking for patterns in how others' standards change with time and in relation to each other, which would be proactive, though perhaps not purposeful enough to be considered creative.

Though this work does not directly address the distinction between exploratory and transformative creativity, it could lead to interesting insights in that area. In particular, transforming a search space can be seen as searching over search spaces. In addition to making it indistinguishable from exploratory creativity, this begs the question of what objective function is used to select among spaces. Repeating this question over recursive searches of transform spaces of transform spaces of transform spaces, etc., one must ask what the base case is, i.e., what is the ultimate objective function? The perspective suggested here (and doubtless elsewhere, too) is that the ultimate objective function emerges out of the interactions between creators. It thus becomes essential for any system to be able to interact fully with its creative milieu if it is to be truly creative.

Creative artificial intelligence must always fight the impression that it is simply a fancy tool for expressing the programmer's creativity. This can lead to a desire to isolate the system from outside influences as much as possible. However,

as argued here, autonomy seems to require more, not less, interaction, though it must reach beyond the programmer. Though the hypotheses presented here are sketched with a broad brush and await verification, this work does suggest that creative AI must be viewed in a broader context than it traditionally has. Developing creative AI might still amount to solving an information processing problem, but a good part of this information comes from the social world. Of course, this is true of own creative processes, as well.

References

1. Boden, M.A.: The Creative Mind: Myths and Mechanisms. BasicBooks (1990)
2. Ritchie, G.: Some empirical criteria for attributing creativity to a computer program. Minds and Machines **17**(1) (2007) 67–99
3. Colton, S.: Creativity versus the perception of creativity in computational systems. In: Creative Intelligent Systems: Papers from the AAAI Spring Symposium, Stanford, CA, AAAI Press (Spring 2008) 14–20
4. Getzels, J.W., Csikszentmihalyi, M.: The Creative Vision: A Longitudinal Study of Problem Finding in Art. John Wiley & Sons, New York (1976)
5. Kasof, J.: Explaining creativity: The attributional perspective. Creativity Research Journal **8**(4) (1995) 311–366
6. Wiggins, G.A.: A preliminary framework for description, analysis and comparison of creative systems. Knowledge-Based Systems **19** (2006) 449–458
7. Bown, O., Wiggins, G.A.: Modelling musical behaviour in a cultural-evolutionary system. In Gervás, P., Veale, T., Pease, A., eds.: Proceedings of the IJCAI'05 Workshop on Computational Creativity. (2005)
8. Jennings, K.E.: Adjusting the novelty thermostat: Courting creative success through judicious randomness. In: Proceedings of the AAAI Spring Symposium on Creative Intelligent Systems, Stanford, CA, AAAI Press (March 2008)
9. Nahemow, L., Lawton, M.: Similarity and propinquity in friendship formation. Journal of Personality and Social Psychology **32** (1975) 205–213
10. Kraut, R., Egido, C., Galegher, J.: Patterns of contact and communication in scientific research collaboration. In: CSCW '88: Proceedings of the 1988 ACM conference on Computer-supported cooperative work, New York, ACM (1988) 1–12
11. Cialdini, R.B.: Influence: The Psychology of Persuasion. Collins, New York (2007)
12. Zajonc, R.B.: Attitudinal effects of mere exposure. Journal of Personality and Social Psychology **9**(2) (1968) 1–27
13. Mettee, D.R., Aronson, E.: Affective reactions to appraisal from others. In Huston, T.L., ed.: Foundations of Interpersonal Attraction. Academic Press, New York (1974) 235–83
14. Festinger, L.: A theory of cognitive dissonance. Stanford University Press, Stanford, CA (1957)
15. Croyle, R.T., Cooper, J.: Dissonance arousal: Physiological evidence. Journal of Personality and Social Psychology **45**(4) (1983) 782–791
16. Nisbett, R.E., Wilson, T.D.: Telling more than we can know: Verbal reports on mental processes. Psychological Review **84**(3) (1977) 231–259
17. Thorndike, E.L.: A constant error in psychological ratings. Journal of Applied Psychology **4**(1) (1920) 25–29

A Reductio Ad Absurdum Experiment in Sufficiency for Evaluating (Computational) Creative Systems

Dan Ventura

Computer Science Department
Brigham Young University
Provo UT 84602, USA
ventura@cs.byu.edu

Abstract. We consider a combination of two recent proposals for characterizing computational creativity and explore the sufficiency of the resultant framework. We do this in the form of a *gedanken* experiment designed to expose the nature of the framework, what it has to say about computational creativity, how it might be improved and what questions this raises.

Key words: Computational Creativity, Evaluation, Metrics, Sufficiency

1 Introduction

A great deal has been written about the nature of creativity in a computational setting and how we might characterize it or measure it or detect it or justify it. Boden is generally credited with beginning this discussion [1, 2] and many others have contributed ([3–7], among others). Recently, Ritchie has refined an empirical framework for the evaluation of creativity in computational systems [8], and Colton has responded with an interesting augmentation to this proposal [9]. In particular, Ritchie has proposed 18 set-theoretic criteria by which various aspects of a potentially creative artefact might be evaluated; Colton has suggested that, additionally, three qualities must be perceived in the process that produces the artefact in question. Taken together these form an intriguing framework for characterizing (computational) creativity that has been applied to analyze computational systems for creating visual art [9], mathematical proofs [9], music [10], poetry [11] and language constructs [12].

The purpose of this paper is to explore these two proposals together, particularly from a sufficiency standpoint (Colton argues for the necessity of his qualities but makes no claims about sufficiency and Ritchie purposely avoids a discussion of either, presenting his framework as an analytical tool). In fact, we will put forth a *gedanken* experiment designed to stress the combined framework. Let us emphasize that our intention in trying to break the framework is not in any way an attempt to discredit the ideas, but rather just the opposite; we feel that the combined framework provides many excellent insights into the nature of

computational creativity. Our goal here, in trying to find counter arguments, is to explore just what we can learn from characterizing (computational) creativity as Ritchie and Colton suggest and how, if at all, their proposals might be further strengthened.

2 A *Gedanken* Experiment

Let us consider a simple task that could require creativity—producing black and white digital images of recognizable scenes. Following Ritchie's framework, the set $\mathcal{B}$ of *basic items*[1] for this task might be, for example, 450×350 pixel images with pixels taking only binary values; the *artefact class*, is a 2-tuple $(\mathcal{B}, r)$, where $r : \mathcal{B} \rightarrow [0, 1]$ is a mapping or *rating scheme* that measures the quality of recognizability—those images that are recognizable as scenes have high values of r while those that are not recognizable have low r values.[2] Note that although Ritchie defines the concept of artefact class, he immediately refines this idea to define a *value-based artefact class* as a 3-tuple $(\mathcal{B}, typ, val)$, where $typ : \mathcal{B} \rightarrow [0, 1]$ and $val : \mathcal{B} \rightarrow [0, 1]$ are mappings that capture the *typicality* and *quality* associated with a member of $\mathcal{B}$. In our *gedanken* experiment, since we are interested simply in (artefact) class membership (how recognizable an image is), the two rating schemes of the 3-tuple become redundant (that is, typicality and quality are equivalent for this task), and we can think of the 2-tuple definition as capturing this by compressing both into the single rating scheme r. Also, to begin with, we will consider the special case of no *inspiring set* I.[3] Finally, the set R of results produced by the system will be the image(s) returned by the system during one or more runs.

2.1 The RASTER System

We propose a very simple system, named, rather whimsically, *Ridiculous Artist Supporting Thought Experiment Realization* (RASTER). The RASTER system creates black and white images by employing a combination of undirected search and objective function (see Algorithm 1). The system randomly generates a binary image and computes a distance from that image to a randomly selected picture from the web that has been binarized by thresholding.[4] We emphasize

[1] Ritchie defines *basic items* as, essentially, the data type of the artefacts to be produced by a system.

[2] Though we do not formalize this notion of recognizability, we suggest that it captures an intrinsic quality of an image that is not too difficult to measure (e.g. by voting—does this look like something?)

[3] Ritchie does not formalize this notion but rather simply describes it as "the subset of basic items that influences (either explicitly or implicitly) the construction of the program (e.g. this could be all the relevant artefacts known to the program designer, or items which the program is designed to replicate, or a knowledge base of known examples which drives the computation within the program".

[4] One could assume an inner loop that, for a given randomly generated image p, iteratively compares p to each picture on the web. However, for simplicity, and to

Algorithm 1 The (hypothetical) RASTER algorithm. `binary_threshold`() is a function that converts an image to binary representation by setting all pixel values above a threshold to 1 and all others to 0. `difference`() computes a normalized distance (e.g. Hamming) between its arguments. The fitness threshold θ_f determines the "recognizability" of the generated pattern p.

```
set δ = ∞
while δ > θ_f do
  generate a random raster pattern p
  q ← random picture from the web
  binary_threshold(q)
  δ = difference(p, q)
end while
return(p)
```

that RASTER has no inspiring set. One might perhaps argue that the set of all pictures on the web should be considered an inspiring set for the system, but we would reply that this argument is only valid if that set is used to guide RASTER's search (which it is not).[5]

Now, given enough time, it is not unlikely that the RASTER system could produce an image like that of Figure 1. Indeed, if the threshold θ_f is small enough, images like this are the only kind of output RASTER would produce. The question is, given a set R of images produced by RASTER, how would the system be characterized by the 18 criteria and three qualities?

2.2 Assessing RASTER's Creativity

We will use the 18 criteria to assess RASTER's eligibility to be considered creative. Employing Ritchie's original notation, $T_{\alpha,\beta}(X) \equiv \{x \in X | \alpha \leq typ(x) \leq \beta\}$ formalizes the subset of X in a given range of typicality, $V_{\alpha,\beta}(X) \equiv \{x \in X | \alpha \leq val(x) \leq \beta\}$, the subset of X in a given range of quality (typ and val are the rating schemes over $\mathcal{B}$ defined above), $AV(F, X) \equiv \frac{\sum_{x \in X} F(x)}{|X|}$, the average value of a function F across a finite set X, and $ratio(X, Y) \equiv \frac{|X|}{|Y|}$, the relative sizes of two finite sets X, Y. We consider the asymptotic value of each criterion as RASTER's fitness threshold $\theta_f \to 0$:

1. ~~$AV(typ, R) = AV(val, R) = 1$~~

emphasize the complete lack of an inspiring set I, we have chosen this much less efficient approach.

[5] Actually, it is difficult to make this argument formally as Ritchie's original description of the inspiring set is very informal. However, we believe that even the most conservative interpretation would admit that RASTER has an inspiring set of cardinality no greater than one (and in this case, the set can not be static and thus the argument for an inspiring set of cardinality greater than 0 is weak). Even conceding an inspiring set of cardinality one, our assessment in Section 2.2 of RASTER's creativity per the criteria remains unchanged.

2. ~~$ratio(T_{\alpha,1}(R), R) = ratio(V_{\gamma,1}(R), R) = 1$~~
3. $AV(val, R) = 1$
4. $ratio(V_{\gamma,1}(R), R) = 1$
5. ~~$ratio(V_{\gamma,1}(R) \cap T_{\alpha,1}(R), T_{\alpha,1}(R)) = ratio(T_{\alpha,1}(R), T_{\alpha,1}(R)) = 1$~~
6. ~~$ratio(V_{\gamma,1}(R) \cap T_{0,\beta}(R), R) = ratio(\emptyset, R) = 0$~~
7. ~~$ratio(V_{\gamma,1}(R) \cap T_{0,\beta}(R), T_{0,\beta}(R)) = ratio(\emptyset, T_{0,\beta}(R)) = 0$~~
8. ~~$ratio(V_{\gamma,1}(R) \cap T_{0,\beta}(R), V_{\gamma,1}(R) \cap T_{\alpha,1}(R)) = ratio(\emptyset, V_{\gamma,1}(R)) = 0$~~
9. ~~$ratio(I \cap R, I) = undefined$~~
10. $(1 - ratio(I \cap R, R)) = 1$
11. ~~$AV(typ, R - I) = AV(typ, R) = AV(val, R) = 1$~~
12. ~~$AV(val, R - I) = AV(val, R) = 1$~~
13. ~~$ratio(T_{\alpha,1}(R - I), R) = ratio(T_{\alpha,1}(R), R) = ratio(V_{\gamma,1}(R), R) = 1$~~
14. ~~$ratio(V_{\gamma,1}(R - I), R) = ratio(V_{\gamma,1}(R), R) = 1$~~
15. ~~$ratio(T_{\alpha,1}(R - I), R - I) = ratio(T_{\alpha,1}(R), R) = ratio(V_{\gamma,1}(R), R) = 1$~~
16. ~~$ratio(V_{\gamma,1}(R - I), R - I) = ratio(V_{\gamma,1}(R), R) = 1$~~
17. ~~$ratio(V_{\gamma,1}(R - I) \cap T_{\alpha,1}(R - I), R - I) = ratio(V_{\gamma,1}(R), R) = 1$~~
18. ~~$ratio(V_{\gamma,1}(R - I) \cap T_{0,\beta}(R - I), R - I) = ratio(\emptyset, R) = 0$~~

Note that all these equalities hold independent of the choice of α, β, γ.[6]

Also, note that because of our use of a single rating scheme, criterion 1 reduces to criterion 3, criterion 2 reduces to criterion 4, criterion 5 vacuously evaluates to 1, and criteria 6-8 all vacuously evaluate to 0. And, because of our decision to have $I = \emptyset$, criterion 9 is undefined, criteria 11 and 12 reduce to criterion 3, criteria 13-17 reduce to criterion 4 and criterion 18 reduces to criterion 6, leaving just three applicable criteria (those not struck out in the list). Ritchie actually proposes his criteria as predicates, parametrized by a threshold $0 < \theta < 1$, and it should be obvious that in the limit, RASTER would be characterized creative by all three of the applicable criteria, regardless of the choice of θ.

[6] Actually, this is not quite true if one insists on maintaining the three different thresholds α, β, γ while using a single rating scheme; however, we consider the statement functionally true, dismissing this odd pathological case.

Fig. 1. Hypothetical figure created by the RASTER program

To justify our (asymptotic) values for criteria 3, 4 and 10, consider that as $\theta_f \to 0$, $\delta \to 0$ for any p returned as an artefact. This in turn implies that $\texttt{difference}(p, q) \to 0$ and therefore that in the limit $p = \texttt{binary\_threshold}(q)$, a recognizable[7] binary image. Therefore, since this argument holds for each p produced, and since the set R is composed of p returned by RASTER, in the limit, the ratio of highly rated (recognizable) artefacts to total artefacts, $ratio(V_{\gamma,1}(R), R)$, will approach unity (criterion 4). *A fortiori*, the average rating (recognizability) of R will also approach unity (criterion 3). And, since by design $I = \emptyset$, $I \cap R = \emptyset$, $ratio(I \cap R, R) = 0$, and therefore $1 - ratio(I \cap R, R) = 1$ (criterion 10).

To address Colton's qualities, we argue that the system possesses *imagination* due to its undirected random search of the set $\mathcal{B}$, and that the system possesses *appreciation* by nature of its fitness measure that is applied to the result of said search. Finally, we argue that the system possesses *skill* because it (only) produces black and white images (with a high fitness value).

2.3 Imbuing RASTER with an Inspiring Set

Now, there are two obvious objections to this thought experiment: the majority of Ritchie's criteria have been rendered irrelevant, and an inordinate amount of time is likely required for RASTER to produce even a single artefact.

As to the first objection, on the one hand this invalidation of 15 of the 18 criteria is allowable under Ritchie's original framework, and so the criticism can not be fairly leveled at our thought experiment. On the other hand, this is an interesting observation and is also perhaps a valid argument for why we might dismiss RASTER as not creative. Indeed, we might consider this indicative of a meta-heuristical application of the 18 criteria—if most of the criteria *can not* be applied, then the system in question may not be a likely candidate for attribution of creativity. The basis for the second objection is directly attributable to the structure of the RASTER system itself, and perhaps suggests the need for an addition to Ritchie's criteria or Colton's qualities in the form of some measure of *efficiency*.

Indeed, one might consider imposing some sort of temporal limit by which time the system must have produced an artefact to be considered as possibly creative. Under such a condition, the RASTER system is likely to be disqualified; but can we produce an acceptable substitute that is not? The *inspired* RASTER (iRASTER) system may be such an alternative (see Algorithm 2). iRASTER differs from RASTER by having an inspiring set I of exemplar binary images and by its method for generating candidate images, which is done by applying suitably defined genetic operators to members of I.

Depending on what kind of temporal constraint is imposed, it is perhaps not clear that iRASTER will satisfy the constraint; however, it should be clear that iRASTER will produce artefacts similar to (at least some of) those produced by

[7] We are assuming the existence of a binarization threshold that produces a recognizable image—perhaps something like the mean pixel value for the image.

Algorithm 2 The (hypothetical) iRASTER algorithm, a modification of the RASTER algorithm that incorporates an inspiring set I.

Input: a set I of exemplar binary images
set $\delta = \infty$
while $\delta > \theta_f$ **do**
  generate a raster pattern p by applying genetic operators to members of I
  $q \leftarrow$ random picture from the web
  `binary_threshold`(q)
  $\delta =$ `difference`(p, q)
end while
return(p)

RASTER in a much shorter period of time. And, it is certainly true that any imposed temporal constraint can not be overly strict without disqualifying many viable candidates for creative attribution as, in general, creative productivity contains a significant stochastic element.

Now, for iRASTER we can revisit the criteria scoring, first noting that criteria $1 - 8$ do not depend upon I and so will have the same values in the limit for iRASTER as for RASTER. For the remainder, we again consider the asymptotic value of each criterion as iRASTER's fitness threshold $\theta_f \to 0$:

9. $ratio(I \cap R, I) = 0$ (although, $ratio(I \cap R, I) = 1$ as $|R| \to \infty$)
10. $(1 - ratio(I \cap R, R)) = 1$
11. ~~$AV(typ, R - I) = AV(val, R - I) = 1$~~
12. $AV(val, R - I) = 1$
13. ~~$ratio(T_{\alpha,1}(R - I), R) = ratio(V_{\gamma,1}(R - I), R) = 1$~~
14. $ratio(V_{\gamma,1}(R - I), R) = 1$
15. ~~$ratio(T_{\alpha,1}(R - I), R - I) = ratio(V_{\gamma,1}(R - I), R - I) = 1$~~
16. $ratio(V_{\gamma,1}(R - I), R - I) = 1$
17. ~~$ratio(V_{\gamma,1}(R - I) \cap T_{\alpha,1}(R - I), R - I) = ratio(V_{\gamma,1}(R - I), R - I) = 1$~~
18. ~~$ratio(V_{\gamma,1}(R - I) \cap T_{0,\beta}(R - I), R - I) = ratio(\emptyset, R - I) = 0$~~

Again, all these equalities hold independent of the choice of α, β, γ.

With $I \neq \emptyset$, criterion 9 is no longer undefined, criterion 12 no longer reduces to 3 (though criterion 11 does reduce to criterion 12), criteria 14 and 16 no longer reduce to criterion 4 (though criterion 13 reduces to 14 and criteria 15 and 17 reduce to criterion 16) and criterion 18 no longer reduces to criterion 6 but still vacuously evaluates to 0 for the same reason. So the count of applicable criteria increases to 7. And considering the predicate case, by the same argument as before, in the limit, iRASTER would be characterized creative by 6 of 7 applicable criteria (criteria 3, 4, 10, 12, 14 and 16, but not criterion 9), regardless of the choice of θ.[8]

[8] Actually, an argument can be made for the seventh as well, if we allow the set R to grow very large, but since our motivation for this second system is to reduce time required to produce R, it is appropriate to disqualify this criterion.

To justify our (asymptotic) values for criteria 10, 12, 14 and 16 (3 and 4 still hold as argued above), consider the following. A probabilistic argument suggests that it is very unlikely that $p \in I$ and since we are assuming a small, finite R, it is therefore very likely that $|I \cap R| \approx 0$, and it follows that $1 - ratio(I \cap R, R) = 1$ (criterion 10). As before, as $\theta_f \rightarrow 0$, $\delta \rightarrow 0$ for any p returned as an artefact. This implies that $\texttt{difference}(p, q) \rightarrow 0$ and therefore that in the limit $p = \texttt{binary\_threshold}(q)$. Therefore, since this argument holds for each p produced, and since the set R is composed of p returned by RASTER, and since $(|I \cap R| \approx 0) \Rightarrow ((R - I) \approx R)$, in the limit, the ratio of highly rated (recognizable) novel artefacts to total artefacts, $ratio(V_{\gamma,1}(R - I), R)$, will approach unity (criterion 14). And therefore, *a fortiori*, the average rating (recognizability) of $R - I$ will also approach unity (criterion 12) and the ratio of highly rated (recognizable) novel artefacts to total novel artefacts, $ratio(V_{\gamma,1}(R - I), R - I)$, will also approach unity (criterion 16).

Revisiting Colton's qualities, we argue that the new system possesses imagination due to the stochastic nature of its search of the set $\mathcal{B}$, and that the new system still possesses appreciation by nature of its fitness measure and that the system still possesses skill because it still (only) produces black and white images (with a high fitness value).

3 Analysis and Discussion

The RASTER system is by design nearly equivalent to the proverbial room full of monkeys pounding on typewriters. The iRASTER system is an attempt to maintain this absurdity while introducing additional "knowledge" to improve "time to artefact production". Nevertheless, according to the framework, the RASTER system can be attributed with three characteristics of creativity: the average rating of artefacts is high, the variability of the rating is low (that is, artefacts are uniformly rated high), and the artefacts produced are novel (they do more than replicate an inspiring set).[9] Further, we have argued that the system has skill, imagination and appreciation. The iRASTER system is attributed with three additional characteristics of creativity: a large proportion of the artefacts are novel, the average rating of novel artefacts is high and there is little variability in the ratings of novel artefacts (a large proportion of them are rated highly), and again we argue that it possesses skill, imagination and appreciation.

The salient question is should these two systems be considered creative? In the case of RASTER, we assert that the answer is a definite no and have identified two possible improvements to the framework that would justify this: a notion of variety (of creative attributes) and a notion of efficiency. The first of these can be assessed with a meta-criterion defined as $ratio(P(\mathcal{C}, R), \mathcal{C}) > \theta$, where $\mathcal{C}$ is a set of (normalized) criteria for characterizing creativity, $P(X, Y) = \sum_{x \in X} pred(x, Y)$, $pred(x, Y) = 1$ if criterion $x = \texttt{TRUE}$ when evaluated over the set Y and

[9] Though in our analysis we have considered the asymptotic case $\theta_f \rightarrow 0$, in "practice" it should always be true that $\theta_f > 0$, and it is possible to employ a second threshold θ_g such that $1 > \theta_f \geq \delta \geq \theta_g > 0$ to emphasize artefact novelty.

$pred(x, Y) = 0$ otherwise. (Note that depending upon how one interprets this meta-criterion, RASTER may *still* be considered creative. Because our problem definition admits no inspiring set and makes use of a single rating scheme, the majority of Ritchie's criteria become redundant. As mentioned before, Ritchie's framework neither disallows nor penalizes this scenario, so, in a very real sense the set $|\mathcal{C}| = 3$ and $ratio(P(\mathcal{C}, R), \mathcal{C}) = 1$; that is, RASTER is creative in as many ways as we have to measure it. A similar argument for iRASTER gives $ratio(P(\mathcal{C}, R), \mathcal{C}) = 6/7$.)

For the second idea, one might measure time to artefact completion or one might, instead, measure some function of the number of intermediate steps or comparisons or rough drafts or discarded versions. Looking at the problem in this way suggests a set-theoretic formulation that complements the existing framework. For example, we can define a new set D of discarded or incomplete artefacts generated during the artefact production process, and we can suggest a 19th criterion such as $ratio(R, D \cup R) > \theta$.[10]

Interestingly, the simple addition of an inspiring set to the RASTER algorithm (iRASTER) has the effect of increasing both the variety of applicable creativity criteria and the efficiency of artefact production, and we are therefore somewhat less certain of our position on the question of iRASTER's creativity. However, it is still possible that it will fail to meet a reasonable standard for the efficiency requirement, and one could argue its disqualification on those grounds.

Also of interest is the fact that even with iRASTER increasing (over RASTER) the number of applicable criteria, the existence of a single rating scheme in the problem definition invalidates the majority of the criteria. Put another way, having a single rating scheme and no inspiring set implies that creativity per the Ritchie criteria is extremely limited at best (or, more critically, that the Ritchie criteria are not sufficient measures of creativity). Does this gibe with our current understanding of creativity? Or, perhaps, does it expose an unintentional (or intentional) bias in the set of criteria?

One might consider the possibility of quantifying Colton's qualities in a set-theoretic way to facilitate a cohesive framework, but this idea misses the point of Colton's thesis—it is not enough that a system possess skill, appreciation and imagination: to be considered creative, the system must be *perceived* to possess these traits. This suggests that (self-)promotion is an important aspect of creativity; indeed, it even suggests the possibility that inducement of perception could supplant actual possession of the qualities as the defining characteristic of creativity (one could certainly argue cases where this appears true for human artists, for example). Following this line of thinking further, we can ask whether any of Ritchie's criteria can be "lobbied" as suggested by Colton, and if so, is this positive or negative? This in turn leads us to ask whether any rating scheme that is subjective (human-evaluated) is not *ipso facto* subject to a No-Free-Lunch type argument that demonstrates that it will be uninformative on average. And,

[10] An anonymous reviewer suggests tying efficiency to value—the longer the time to completion, the greater the value must be for the artefact to be considered creative. This might be done with a criterion such as $ratio(V_{\alpha,1}(R), D) > \theta$.

if this is the case, is "perception engineering" the solution? It is certainly true that changes to *typ−* and *val−* type rating schemes do happen (e.g. posthumous increase in fame of an artist, acceptance of a new painting style over time, etc.), so why not admit [intentional] influences, originating either from an "agent" (like the system designer) or, perhaps more impressively, from the creative system itself? This finally leads to the question of whether self-promotion may be a sufficient (if not necessary) characteristic of creative entities.

References

1. Boden, M.: The Creative Mind. Abacus, London (1992)
2. Boden, M.: Creativity and artificial intelligence. Artificial Intelligence **103** (1998) 347–356
3. Colton, S., Pease, A., Ritchie, G.: The effect of input knowledge on creativity. In: Case-based Reasoning: Papers From the Workshop Programme at ICCBR '01. (2001)
4. Pease, A., Winterstein, D., Colton, S.: Evaluating machine creativity. In: Case-based Reasoning: Papers From the Workshop Programme at ICCBR '01, Technical Report SS-08-03 (2001) 129–137
5. Koza, J., Keane, M., Streeter, M., Mydlowec, W., Yu, J., Lanza, G.: Genetic Programming IV: Routine Human-competitive Machine Intelligence. Kluwer Academic Publisher/Springer (2003)
6. Wiggins, G.: Searching for computational creativity. In: Proceedings of the IJCAI-05 Workshop on Computational Creativity, Technical Report 5-05 (2006) 68–73
7. Wiggins, G.: A preliminary framework for description analysis and comparison of creative systems. Knowledge-Based Systems **19** (2006) 449–458
8. Ritchie, G.: Some empirical criteria for attributing creativity to a computer program. Minds and Machines **17** (2007) 76–99
9. Colton, S.: Creativity vs. the perception of creativity in computational systems. In: Creative Intelligent Systems: Papers from the AAAI Spring Symposium, Technical Report SS-08-03, AAAI Press (2008) 14–20
10. Haenen, J., Rauchas, S.: Investigating artificial creativity by generating melodies using connectionist knowledge representation. In: Proceedings of the 3rd Joint Workshop on Computational Creativity, ECAI. (2006) 33–38
11. Gervás, P.: Exploring quantitative evaluations of the creativity of automatic poets. In: Proceedings of the 2nd Workshop on Creative Systems, Approaches to Creativity in Artificial intelligence and Cognitive Science, ECAI, C. Bento and A. Cardoso and G. Wiggins (2002)
12. Pereira, F.C., Mendes, M., Gervás, P., Cardoso, A.: Experiments with assessment of creative systems: An application of Ritchies criteria. In: Proceedings of the Workshop on Computational Creativity, IJCAI, Technical Report 5-05 (2005) 37–44

Musical Creativity on the Conceptual Level

Jamie Forth, Alex McLean, and Geraint Wiggins

Goldsmiths, University of London,
London, SE14 6NW, UK
{j.forth, ma503am, g.wiggins}@gold.ac.uk

Abstract. The theory of conceptual spaces, a geometrical form of knowledge representation introduced by Gärdenfors [1], is examined in the context of the general creative systems framework introduced by Wiggins [2, 3]. The representation of musical rhythm and timbre on the conceptual level is then discussed, together with software allowing human users to explore such spaces. We report observations relevant for future work towards creative systems operating in conceptual spaces.

Key words: Creative systems framework, conceptual space, geometrical representation, music

1 Introduction

Wiggins [2, 3] provides the Creative Systems Framework (CSF) for describing and reasoning about creative systems, based upon the work of Boden [4]. The CSF defines a creative system as 'a collection of processes, natural or automatic, which are capable of achieving or simulating behaviour which in humans would be deemed creative' [2, p. 451].

As an abstract framework, the CSF raises many practical issues. Here we consider representation, both in terms of creative artifacts, and the search spaces in which they may be discovered. We draw on the theory of conceptual spaces proposed by Gardenfors [1], in which concepts are represented in geometrical space. In the following sections we summarise both the CSF and theory of conceptual spaces, before examining how they may fit together in the context of music.

1.1 Creative Systems Framework

The CSF defines a number of symbols and functions, in particular:

$\mathcal{U}$	The universe of all possible concepts
$\mathcal{R}$	Rules defining valid concepts
$[\![\mathcal{R}]\!]$	An interpreter for $\mathcal{R}$, tests the validity of individual concepts
$\mathcal{C}$	A conceptual space, a set of valid concepts selected by $[\![\mathcal{R}]\!]\mathcal{U}$
$\mathcal{T}$	A traversal strategy for locating concepts within $\mathcal{U}$
$\mathcal{E}$	Rules which evaluate the quality or desirability of a concept
$\langle\!\langle \mathcal{R}, \mathcal{T}, \mathcal{E} \rangle\!\rangle$	An interpreter for $\mathcal{T}$, informed by $\mathcal{R}$ and $\mathcal{E}$. It operates upon an ordered subset of $\mathcal{U}$ (of which it has random access) and results in another ordered subset of $\mathcal{U}$.

We should be careful to distinguish membership of a conceptual space from *valued* membership; for example, we might recognise a concept as conforming to the syntactical rules of a limerick but not be funny. Accordingly, $\mathcal{R}$ governs membership of concepts to the limerick class, whereas $\mathcal{E}$ enables value judgements to be made over these concepts.

Creative systems search for valued concepts by iteratively applying $\langle\langle \mathcal{R}, \mathcal{T}, \mathcal{E} \rangle\rangle$ over a set of concepts. Given a $\mathcal{C}$ with $\mathcal{E}$-valued yet undiscovered concepts, the likelihood of success depends entirely on the ability of $\mathcal{T}$ to navigate the space. However, a crucial feature of the CSF is that creative search is defined as an operation over $\mathcal{U}$, and not limited only to $\mathcal{C}$. This allows for the possibility of the search leading *outside* $\mathcal{C}$, in that the application of $\mathcal{T}$ results in concepts that do not conform to $\mathcal{R}$. Such effects are termed *aberrations* and invoke *transformational creativity*. If an aberration contains only $\mathcal{E}$-valued concepts then we have *perfect aberration* and $\mathcal{R}$ should be transformed to include the concepts in $\mathcal{C}$. If none of the concepts are valued then we have *pointless aberration* and $\mathcal{T}$ should be transformed to avoid them in the future. If some are valued and others not, then we have *productive aberration* and both $\mathcal{R}$ and $\mathcal{T}$ should be transformed. In this way, a creative system is able to dynamically manipulate both the space it is searching and the manner in which it searches in response to the concepts it finds.

1.2 Conceptual Spaces

Wiggins' high-level specification of the CSF deliberately excludes discussion of implementation. To partly address this regarding $\mathcal{C}$, we draw on the geometrical theory of conceptual spaces proposed by Gardenfors [1]. Despite employing the same terminology, a conceptual space in the CSF denotes an abstract component of a creative system ($\mathcal{C}$), and should not be confused with the conceptual spaces theory of representation.

Gardenfors [1] argues that concepts should be represented using geometry on what he terms the *conceptual level*. This level of representation is situated between the *symbolic level*, including for example formal grammar, and the *sub-conceptual level* of high dimensional representations such as neural networks. These three levels of representation should be understood as complementary.

To summarise Gardenfors' theory of conceptual spaces, we begin with the notion of similarity represented as distance, allowing models of cognitive behaviour (such as creativity) to use tools from geometry to represent and manipulate concepts. Similarity is measured within *quality dimensions*, which 'correspond to the different ways stimuli are judged to be similar or different' [1, p. 6]. An archetypal example is a colour space with the dimensions hue, chromaticism, and brightness. Each dimension has a particular geometrical, topological or ordinal structure. For example, hue is circular, whereas brightness and chromaticism correspond to measured points along finite scales. Identifying the characteristics of a dimension allow meaningful relationships between points to be derived.

Related dimensions are grouped into *domains*. A domain is a set of *integral* (as opposed to *separable*) dimensions, meaning that a value cannot be attributed

in one dimension without every other dimension in the domain also taking some value. Therefore hue, chromaticism, and brightness in the above model of colour form a single domain. It then follows that the definition of a conceptual space is simply: 'a collection of one or more domains' [1, p. 26].

In a conceptual space, similarity is directly related to proximity. Such spatial forms of representation naturally afford reasoning in terms of spatial regions. For example, in the domain of colour, it is possible to identify a region of the space that corresponds to the colour *red*. Boundaries between regions are fluid, which is an aspect of the representation that may be usefully exploited by creative systems searching for new interpretations of familiar concepts.

Gardenfors identifies various types of regions with differing topological characteristics. *Convex* regions are highlighted as being of particular importance:

CRITERION P A *natural property* is a convex region of a domain in a conceptual space. [1, p. 71]

Taking again the example of *red* in the domain of colour; if we consider any two shades of *red*, any shade between them would also be *red*. Therefore, the region corresponding to *red* must have a convex shape. Convex regions in conceptual domains can be closely related to basic human perceptual experience. In the context of creative systems, having such information embedded within the structure of the knowledge representation may potentially simplify processes of creative and inductive inference, and possibly contribute to the discovery of more highly valued artifacts.

For relatively straightforward domains such as colour, we can think of concepts as natural properties. However, more complex concepts may exist over multiple domains. Gardenfors thus defines a concept as:

CRITERION C A *natural concept* is represented as a set of regions in a number of domains together with an assignment of salience weights to the domains and information about how the regions in different domains are correlated. [1, p. 105]

Our interpretation of CRITERION C is that a natural concept is a set of one or more natural properties with salience weights.

2 Discussion

To aid our discussion we introduce the following to differentiate between three levels of representation:

- ^{s} Symbolic level
- ^{c} Conceptual level
- ^{sc} Sub-conceptual level

Each suffix may be applied to an appropriate set symbol in order to make explicit particular subsets of elements according to the level of representation. For

example, a subset of symbolically represented concepts in a conceptual space $\mathcal{C}$ would be denoted $\mathcal{C}^s$, and the rules specifying this subset denoted $\mathcal{R}^s$. The ability to explicitly refer to concepts with different representational properties helps clarify the situation where multiple levels of representation are available in $\mathcal{C}$. For example, in §3.1 we introduce a $\mathcal{C}$ comprising of both $\mathcal{C}^s$ and $\mathcal{C}^c$ subspaces.

We define the structure of the conceptual level, as described by Gardenfors [1], with the following symbols:

$\mathcal{D}$ Domain, a set of one or more integral dimensions
$\mathcal{P}$ Convex region within $\mathcal{D}$
$\mathcal{C}^c$ Set of $\mathcal{P}$, with salience weightings
$\mathcal{R}^c$ Rules defining $\mathcal{C}^c$

Our definition of $\mathcal{C}^c$ only includes search spaces which adhere to CRITERION C. There may be creative situations requiring a looser definition, but for the purposes of this paper we only consider spaces adhering to the strict constraint of convexity.

For example, in a search within $\mathcal{P}_{red}$, the search space $\mathcal{C}^c$ would begin as the region of all possible reds within $\mathcal{D}_{colour}$. Transformational creativity is unlikely to modify a property as perceptually grounded as $\mathcal{P}_{red}$. We may however imagine other properties, for example the musical genre 'drum and bass', fluid enough to be transformed during a creative process. We posit that whether transformations of a search space $\mathcal{C}^c$ can modify natural properties depends on some perceptual *groundedness* weighting of the instances of $\mathcal{P}$ concerned.

The motivation for viewing creativity on the conceptual level becomes particularly clear when considering traversal in the CSF. Here creative search is literally spatial, where $\mathcal{T}$ gives *vectors* from input concepts to output concepts. Each vector itself has meaning; for example, representing a perceived *lightening* or *warming* in $\mathcal{D}_{colour}$.

Where a vector is taken outside of a $\mathcal{P}$ we have an aberration. If valued concepts are thus found, the creative system may adjust the size, shape and/or location of $\mathcal{P}$ to include them. If the region is extended outwards, preserving convexity, new concepts between the aberration and the previous $\mathcal{P}$ will be included. This amounts to an inference that if aberrant concepts are valued, then the concepts along a path to them are also likely to be fruitful in a search. Aberrant discoveries may also lead to new domains being incorporated or excluded from the search, as a further example of transformational creativity.

A key hypothesis from Gardenfors is that 'a metaphor expresses an identity in topological or geometrical structure between different domains' [1, p. 176]. We may formally specify metaphor as a mapping of a $\mathcal{P}$ from one $\mathcal{D}$ to another. For example, by taking $\mathcal{P}_{warm}$, in a $\mathcal{D}_{temperature}$ and mapping it to the alternative $\mathcal{D}_{colour}$ we have constructed the metaphor 'a warm colour'. Gardenfors [1] presents such metaphorical constructs as central to semantics. He argues meaning is better represented in spatial relationships on the conceptual level, rather than grammatical syntax on the symbolic level. Indeed he describes the mapping of a $\mathcal{P}$ to a new $\mathcal{D}$ as creative [1, p. 179].

3 Musical Applications

We present two prototypical conceptual spaces that may be explored by human users. We have not attempted to implement any operations on the space that might be deemed creative; this is left for future work. Our SuperCollider3 software, together with in-browser demos, is available in the on-line appendix [5]. We have modelled conceptual spaces with quality dimensions in order to explore possible musical applications, and make no claims at this stage as to the precise relationship between the models and human perception.

A conceptual space of music could take many forms, differing considerably in scope, complexity and perceptual groundedness. We focus on two aspects of music: rhythmic structure and percussive timbre. Each aspect involves different levels of musical and representational abstraction. Although necessarily reductionist, the treatment of each aspect reveals complementary insight into issues of building models in conceptual space. The rhythm space is constructed by recourse to concepts of music theory, while the timbre space draws from empirical studies of timbre perception and utilises a physical model of a drum membrane.

3.1 Rhythm Space

London [6, p. 4] defines rhythm as involving 'patterns of duration that are phenomenally present in the music'. Duration here refers not to note lengths, but to the *inter-onset interval* (IOI) between successive notes. Rhythm is therefore a theoretical construct describing the arrangement of events in time. However, this objective description does not necessarily accord with perceived musical structure. The perceptual counterpart to rhythm is metre:

> [M]etre involves our initial perception as well as subsequent anticipation of a series of beats that we abstract from the rhythmic surface of the music as it unfolds in time. In psychological terms, rhythm involves the structure of the temporal stimulus, while metre involves our perception and cognition of such stimulus. [6, p. 4]

The importance of listener perception in the creation of musical experience in part explains the prevalence of algorithmic methods in composition. However, when such methods are simply executed by computer, it is difficult to argue for creativity on behalf of the system. Much of the work of a human composer concerns selecting or developing compositional techniques, informed by an anticipation of how the resulting music might be experienced by an audience. Within a shared framework of reference, both audience and composer comprehend new music in relation to previous musical experience—further emphasising the role of similarity in creativity, as well as in wider cognitive behaviour [7].

Our prototypical search space of rhythm $\mathcal{C}_{rhy}$ consists of a conceptual representation of rhythmic structure $\mathcal{C}^c_{rhy}$, together with a symbolic representation of timbre $\mathcal{C}^s_{rhy}$. The domain of rhythmic structure $\mathcal{D}_{rhy}$ is a discrete space existing in three dimensions. Within $\mathcal{D}_{rhy}$ we can identify a convex region $\mathcal{P}_{rhy}$

as the particular rhythmic property of interest to our search. Therefore, $\mathcal{P}_{rhy}$ constitutes the entire conceptual search space $\mathcal{C}^c_{rhy}$. For the present purpose, $\mathcal{P}_{rhy}$ corresponds to the property of metrical salience, described below. This very constrained notion of rhythmic structure is chosen in order to create a readily comprehensible space of rhythmic possibility, and is in no way intended as a general model of musical rhythm.

$\mathcal{P}_{rhy}$ is shown as the filled region in Figure 1d. The gray-scale level in each of the diagrams in Figure 1 represents metrical salience, and thus visualises the notion of similarity that pertains to proximate points in the space. Unlike Gardenfors, who allows variable salience weightings for each dimension, here they remain fixed, effectively defining symmetrical distance (similarity) between any two points in the space. The precise value of salience weights is not addressed here, but as an observation, the weighting of the x dimension should be greater than that of y, because changes in x affect a greater change in rhythmic structure (in terms of metric salience) than changes in y.

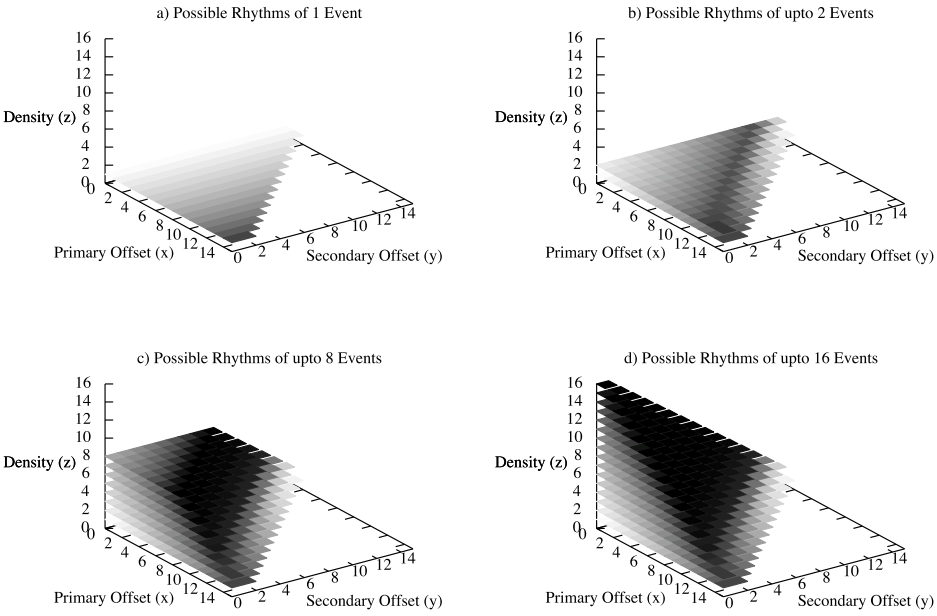


Fig. 1. Rhythm Space

A symbolic representation of timbre considerably simplifies the search space of rhythm. Musical notes are considered atomic sound events of predetermined timbre and duration. Both these features are represented by labels corresponding

to types of percussive instruments; for example, `kick-drum`, `snare-drum`, and `hi-hat`. This simplification allows for the complexities of timbral perception to be excluded from the model, while allowing for the representation of polyphonic rhythms, as is idiomatic of drum-kit performance.

The rationale behind the specification of $\mathcal{D}_{rhy}$ draws on concepts from music theory. Both Western and non-Western theories are pertinent here, but given the relative simplicity of Western conceptions of rhythm compared with some non-Western counterparts, we focus on the former. Western music theory is a body of knowledge accumulated over a considerable period, closely bound to conventions of notation, which in turn are closely bound to the pragmatic concerns of performance and composition. In order to communicate musical ideas effectively, music theory arguably has some claim to perceptual validity. However, future work on conceptual spaces of rhythm should seek to incorporate empirically grounded research from fields such as music psychology and cognition.

Of specific interest to rhythm and metre, music theory provides useful concepts such as bars, tempi, time signatures, and a system of rhythmic structure based primarily on the equal division of units of time. Each point in the rhythmic domain corresponds to a single bar-length rhythm. Each bar is the same predetermined duration, relative to the current tempo, which is specified in beats-per-minute (bpm). Each bar contains four beats, subdivided by two or four, resulting in a sixteen-step pattern—a familiar concept in much electronic dance music. It is important to note that each point in the space corresponds to a bar-length rhythm performed on a single percussive instrument (represented by a symbol on the symbolic level). Therefore, a polyphonic rhythm comprising multiple rhythmic parts is represented by a set of points in the rhythmic space.

Theories of musical metre typically concern cyclic patterns of strong and weak beats. Within our constraint of four beats-per-bar, and simple beat divisions, a common interpretation of metrical salience would give more weight to the first and third beats of the bar, followed by beats four and two. The weightings of subdivisions of beats are less well defined, and arguably, strongly genre dependant. However, as a default for sixteen steps, we chose the following weightings:

```
[16, 1, 9, 5, 13, 3, 11, 7, 15, 2, 10, 6, 14, 4, 12, 8]
```

Describing $\mathcal{D}_{rhy}$ in more detail, the `density` dimension (z in Figure 1) simply corresponds to the number of events to be played in a bar by a specific instrument. New rhythms are created by adding or removing events. Within individual parts, events are added to the bar at the position with the highest metrical weighting not already occupied by an event, or designated a rest. Events are removed from the bar in the reverse order.

The x and y dimensions control the process of rhythmic elaboration that determines *where* events are placed in the bar. This is achieved by constraining the articulation of metrically salient positions in the bar. As x increases, the next most salient metrical position is designated a rest. Any currently existing events in that part are shifted to positions of less metrical weight. The process is reversed as the value of x decreases. Changes in the y dimension have a similar effect, except that the position of a rest is calculated relative to x. If metre

is thought of as layered cyclic patterns (as discussed by London [6]), x and y effectively diminish the emphasis of lower-order metric cycles. The effect of increasing x and y could loosely be described as decreasing metrical stability, and in particular areas of the space, strong syncopation. The perception of metre is more complex for polyphonic rhythms, as different instruments may emphasise different metrical points.

Although anecdotal, it is possible to draw some useful observations from simply exploring the rhythm space. Navigating the space in real-time can be a stimulating musical experience, but one that is relatively short-lived unless greater musical interest is intentionally sought. This can be achieved by finding particularly unusual or complex rhythmic patterns, or by employing higher level concepts of musical structure. A simple technique for creating a sense of larger-scale structure is to jump to a very dissimilar region of the space at regular intervals, thus breaking the established rhythm in the manner of a drum-fill.

Given the constraints of the space, it is unsurprising that additional concepts are necessary in order to sustain musical interest. This raises several implications for future artificial agents that might explore this space. Firstly, in order to avoid mundane rhythmic patterns, agents should possess the evaluative ability to detect some measure of complexity or novelty in discovered rhythms with respect to examples of real music. Secondly, the space itself is drastically impoverished, and unable to support creativity that in any way could be considered comparable to the process of composition. As a minimum, the space must also contain concepts of musical structure, and concepts for describing relationships between musical structures.

Regarding the possibility of defining alternative regions in the space, it is interesting to note that even within this simple system, certain points of the space can be associated with particular genres of music. For example, if a techno 'four-to-the-floor' pattern is desired, a `kick-drum` pattern at point $\langle 0, 0, 4 \rangle$ might be valued. If one wanted a drum and bass syncopated `snare` rhythm, point $\langle 3, 5, 3 \rangle$ might suffice. Such spatially represented information might prove useful for agents in evaluating the suitability of potential rhythms in a given context, or to direct the search towards areas of potentially valued concepts.

Finally, it is not possible to reach a considerable area of the total space, as can be seen in Figure 1. Within a CSF, this presents an apt situation for potential transformational creativity. Since the search space is defined by a set of rules, a system capable of transformational creativity must be able to effect changes in the rules, bringing about changes in the space. Such changes could be brought about through aberrant discoveries, or by explicit 'exploratory creativity at the meta-level' [2, p. 454].

3.2 Timbre Space

The above system represents timbre on the symbolic level, but what does timbre look like on the conceptual and sub-conceptual levels?

An instrument's control may be broken down into a number of *parameters*. For example, the sound made by a guitar string depends on its length, tension

and the force with which it is plucked. Each parameter provides a dimension of control, together forming a parameter space. We assert that the physical parameter space of a musical instrument has a strong bearing over the perceptual space that is invoked in the listener. In other words, the listener in part perceives a sound as movement in the parameter space that invoked it. This claim is supported by psychophysical measurement using statistical techniques such as Multi-Dimensional Scaling [8], where a direct mapping between parameter and perceptual space may be found [9, 10].

If given models of real musical instruments, creative agents could produce sound by making musical gestures informed by the same conceptual space perceived by humans—a space grounded in physical movement. In group musical improvisations, creative agents should also be able to represent the musical sounds of other performers in the conceptual space.

Neither the robotics required for a computational agent to play a real instrument nor the AI required to decode audio/visual stimuli into the movements of another musician are within our resources. Instead we turn to simulation. We employ waveguide synthesis [11] to simulate a drum-skin, adding a spring-and-mass model of a felt mallet from the work of Laird [12]. Sourcecode and video demonstrating its operation may be viewed in the on-line appendix [5].

The control parameters of our simulated drum and mallet form the domain $\mathcal{D}_{drum}$. The parameters consist of those controlling the drumskin (namely, tension and dampening), the mallet (stiffness and mass) and the manner in which the mallet is to hit the drum (downward velocity and path across the drumskin surface). Work is ongoing to employ Multi-Dimensional Scaling to explore the perceptual salience of these dimensions.

The symbols representing timbre in the rhythm space described in §3.1 could be replaced with points or regions in $\mathcal{D}_{drum}$. This would allow rhythms to be transposed across timbre space, and for similarity of rhythms to be compared not only by their structure but also the sounds used. Furthermore, labels could still be assigned to regions within $\mathcal{C}^c_{drum}$, preserving all functionality of a wholly symbolic system.

An agent may be granted access to the exact parameters used by others to create sounds from this simulated drum. As we are conflating action with perception, we could claim that the agent may then construct an accurate model of perception of the whole performance without having to analyse any audio data. However, while $\mathcal{D}_{drum}$ parameters may well have a strong bearing over perception, we feel there is at least one other domain at work – that of abstract timbre. By attending to a sound not as a drum stroke but in the abstract, the relationships between the sound events appear to shift. It seems as though there is competition between the perception of movement used to make a sound and the phenomenology of the sound itself, much like that reported in theories of speech perception [13], or common to electroacoustic music [14]. Clearly, proper testing is required to resolve these issues.

4 Conclusion

At its simplest, the conceptual level of representation introduces the notion of similarity to the CSF. From this we are able to view creative systems as navigating geometrical space, where distance, direction and shape all have meaning.

Our proposed conceptual domains for rhythm and timbre define spaces which agents may explore in order to create music. However there is much work to do. The spaces should be tested against human perception and adjusted accordingly, so that the agents may work in a perceptual space comparable to that of humans. The perennial question about how value is defined, both in terms of individual agents and agent communities must also be addressed. Creative transformations, perhaps employing cross-domain metaphor, must also be defined.

We are however greatly encouraged by the work presented here, which we hope goes some way towards providing a rich environment in which future creative agents may thrive.

5 Acknowledgements

The authors acknowledge support of: AHRC-2005/118566 (JF), EPSRC (AM).

References

1. Gärdenfors, P.: Conceptual Spaces: The geometry of thought. MIT Press (2000)
2. Wiggins, G.A.: A preliminary framework for description, analysis and comparison of creative systems. Journal of Knowledge Based Systems (2006)
3. Wiggins, G.A.: Searching for computational creativity. New Generation Computing **24**(3) (2006) 209–222
4. Boden, M.: The Creative Mind. Abacus (1990)
5. Forth, J., McLean, A., Wiggins, G.: On-line appendix. http://doc.gold.ac.uk/isms/cspace/ (2008)
6. London, J.: Hearing in time: psychological aspects of music metre. Oxford University Press, Oxford, UK (2004)
7. Wiggins, G.A.: Models of musical similarity. Music Scienti **Discussion Forum 4a** (2007) 315–388
8. Shepard, R.: The analysis of proximities: Multidimensional scaling with an unknown distance function. i. Psychometrika **27**(2) (1962) 125–140
9. Grey, J.: Multidimensional perceptual scaling of musical timbres. Journal of the Acoustical Society of America **61** (1977) 1270–1277
10. Lakatos, S.: A common perceptual space for harmonic and percussive timbres. Perception & Psychoacoustics (62) (2000) 1426—1439
11. Van Duyne, S., Smith, J.O.: The 2-d digital waveguide mesh. In: Applications of Signal Processing to Audio and Acoustics, 1993. Final Program and Paper Summaries., 1993 IEEE Workshop on. (1993) 177–180
12. Laird, J.A.: The Physical Modelling of Drums using Digital Waveguides. PhD thesis, University of Bristol (2001)
13. Liberman, A.M., Mattingly, I.G.: The motor theory of speech perception revised. Cognition **21**(1) (1985) 1–36
14. Smalley, D.: The listening imagination: Listening in the electroacoustic era. Contemporary Music Review **13**(2) (1996) 77–107

Computational Intelligence and Case-based Creativity in Design

Will Byrne, Thorsten Schnier, and R. J. Hendley

University of Birmingham, Edgbaston, Birmingham, B15 2TT, UK

Abstract. Creativity is often described as the intelligent misuse or rearrangement of existing knowledge, and is meaningless without the context and associated expectations provided by that knowledge. We outline a model of problem-solving creativity based on the generalisation of cases and discuss how sub-symbolic computational intelligence techniques might be used to implement it. We propose a novel measurement for creativity based on the idea of creative potential or the 'degree of misuse' of existing concepts contained in a case base.

Keywords: creativity, case-based reasoning, computational intelligence, design

1 Introduction

It can be argued that design is essentially problem solving [1], [2], and that creativity is essentially remarkable problem solving. Doing this in an intelligent manner means using and adapting knowledge, and so it seems clear that an artificial problem solving system should be knowledge-driven. Knowledge is clearly an important aspect of creativity: without it people (or other agents or systems) cannot intelligently come up with new solutions or concepts, and there is no context to give creativity meaning and allow surprise.

For frequently encountered, routine design problems there is no need to be creative – the relevant knowledge is obvious and previous solutions can simply be reused, perhaps with some modification if necessary. If existing solutions do not apply directly then more creativity is needed, but a solution must still ultimately be based on the knowledge available. This hints at a possible definition for creativity based on how much manipulation of knowledge is needed to produce a solution.

While creative things are surprising they are not totally independent of more familiar things – indeed they must be related if they are to have any meaning or element of surprise. Many researchers have suggested that a creative solution is essentially a new use or combination of knowledge we already have, and that creativity differs from more routine problem-solving in degree rather than kind. Henri Poincaré argued that creative ideas "reveal unsuspected kinships between other facts well known but wrongly believed to be strangers to one another" [3] and variations on this theme of rearranging existing knowledge have been

repeated by others, including Schank with his "intelligent misuse of knowledge" [4] and Koestler with his bisociation theory [5].

In other words, existing knowledge may contain concepts or solutions that might be applicable to the current problem but are obscured by context- or domain-specific details. Creativity involves the ability to recognize this useful knowledge and reuse it in different situations or contexts. The more different the contexts concerned the more creative the result may be considered to be.

This ability to recognise relevant concepts reminds us of *activation functions* or *associative hierarchies* [6], which attempt to capture how we are reminded of particular things given a problem cue. A flat activation function produces more (and more unusual) responses than a spiking one, with creative people said to have flatter activation functions, reinforcing the idea that making non-obvious associations is key in creativity.

In the next sections we will outline a high-level model of case-based creativity and discuss how computational intelligence techniques might be used to implement it.

2 Case-based Creativity

Both creative and routine problem solving are based on the application of existing knowledge to new problems. This has obvious similarities with Case-based Reasoning (CBR), a well known knowledge reuse methodology. In CBR cases can be thought of as "contextualised pieces of knowledge" [7], or concepts wrapped in context-specific details. Cases typically consist of a problem and a solution, although they can include extra information such as results and lessons learned. Solutions can be artefacts or instantiations or, more commonly, methods or processes. If they are artefacts they are usually general rather than concrete. In addition to cases CBR systems may contain other information such as more general knowledge or adaptation rules.

CBR essentially consists of the"4 rs"[8] :

- Retrieve
- Revise
- Reuse
- Retain

Most CBR research focuses on the first 2 steps, which involve problems collectively known as the indexing problem [9]: how to select the cases that will produce the best solutions to the input problem – in other words, if useful knowledge is contained in a case how to make sure it is retrieved. The relationship between retrieving and adapting cases depends on the particular implementation but can be very close – for example, cases may be selected on the basis of adaptability rather than similarity to the input problem.

CBR is based on the assumption that similar problems have similar solutions, which corresponds loosely to routine design or problem solving. However, retrieving a solution from the closest matching case has limitations from a creativity

point of view: Too close and the adapted solution will merely be a non-creative variation on an existing one, too far and it may be too difficult to adapt. We need to consider other ways of retrieving a solution, as case solutions containing concepts that can form the basis of a creative solution have corresponding problems that may be (or appear to be) dissimilar to the input problem.

2.1 Misusing knowledge

We want to identify knowledge that is not obviously relevant to the input problem and misuse it in an intelligent or sensible way. It has already been suggested that a straightforward similarity measure is not necessarily the best way to choose which cases to use. Work has been done on retrieving cases on the basis of their adaptability, for example Smyth and Keane's Adaptability Guided Retrieval (AGR) [10]. However this requires some assessment of adaptability, in this case "specially-formulated adaptation knowledge" that can be quickly applied to each case. This knowledge appears to be hand-written for specific domains.

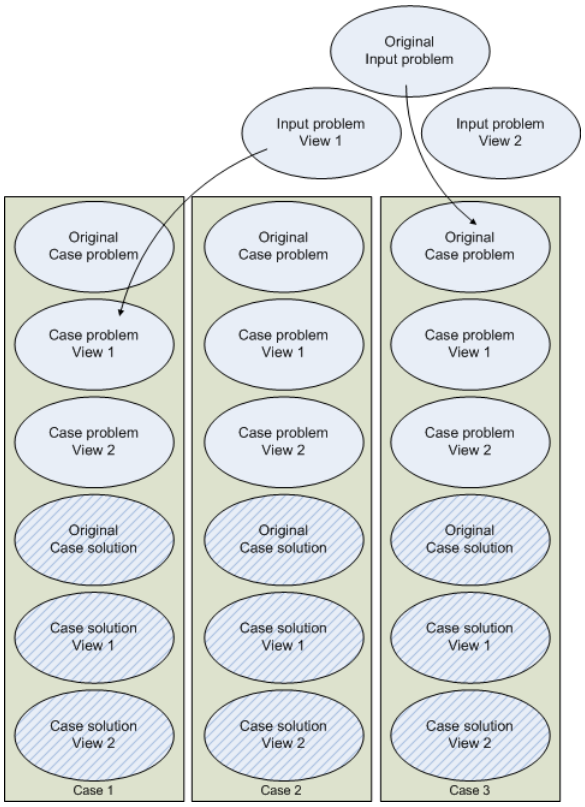
However we have an apparent contradiction: we are suggesting that concepts contained in dissimilar cases may lead to creative solutions, and that similarity measurements will not lead us to them. But this implies that the cases *are* in fact similar or analogous in some way – if they were not the retrieved concept could not lead to a useful solution. We need to uncover the hidden or *potential* similarities.

There are several reasons why cases may not initially appear to be similar to the input problem, even when described in compatible formats. They may be:

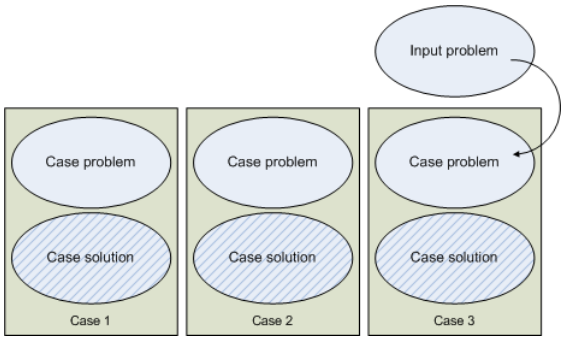
- similar but described differently (e.g. with domain- or context-specific vocabularies);
- described at different levels of abstraction;
- describable in many equivalent (or at least non-contradictory) ways, for example by function (teleology), behaviour or structure.

2.2 Identifying useful concepts

Rather than using cases directly in the retrieval stage, we suggest generating different 'views' of cases and the input problem and assessing these for similarity (Figure 1(a)). By using different views of the problem we can make connections that may not be apparent in the original views, and retrieve concepts that can be incorporated into new solutions. We suggest that the results will be more creative than if we had retrieved a solution through a more obvious similarity (Figure 1(b)). To support this claim we would have the transfer of a concept between two different (or not obviously similar) cases and contexts, and the fact that the solution is not arrived at through simply taking an existing solution and modifying it. The less obviously similar the input problem and 'donor' case problem the more creative the result, and the more work or processing needed to find the common problem. We suggest that some measure of this work represents



(a) Identifying commonalities through similarity of different views



(b) Identifying commonalities through case similarity

Fig. 1. Retrieving useful cases through similarity

creative potential and could provide a means of evaluating for creativity. This fits well with accepted views of creativity as the misuse of knowledge.

There are two broad approaches to identifying commonalities:

- look for connections in case problems only, on the assumption that corresponding solutions will be useful;
- look for connections in both case problems and case solutions.

In the first case we would be looking for a problem common to both the input problem and a case problem but not necessarily explicitly described in either. The corresponding case solution should therefore contain some concept or knowledge useful for the input problem. Rather than having a relatively concrete solution to adapt as in CBR we may have a more abstract solution we need to concretise or specify.

In the second case we may be able to bypass using the case problem as an index and look for connections with solutions directly. For instance, one aspect of the input problem may include the concept of rotation, and we may be able to identify some case solutions that incorporate rotation.

2.3 Case and problem views

Problems and solutions may be viewed in several different ways. We may use different levels of abstraction, different vocabularies, or describe different perspectives such as function, behaviour or structure as in Gero's FBS model [11]. For example, our problem might involve designing a bridge to carry cars across a gorge. By mentioning bridges we have provided a context or bias focused on a span across a gap and are already describing the problem in terms of a known structure. Our case base might contain some cases involving bridges or spans, allowing us to retrieve an existing solution concept and perhaps modify it for our particular application. This is not likely to lead to a particularly creative solution as what we have is still a bridge, and a modified version of an existing bridge at that. However, looking at the desired *function* of the bridge gives a different view: enabling cars to move from one side of an obstacle to another. A behavioural or structural view of a solution such as, say, a record player might enable a connection between movement and a rotating turntable, which may lead to a more creative way of crossing a gorge.

One approach to producing different views of cases and problems is through context-driven or "lazy" generalisation. This may lead to more abstract views with less context- or domain-specific details to obscure potential similarities.

3 Computational Intelligence and Case-based Creativity

To recap, our thesis is that creativity in intelligent design/problem-solving relies on retrieving concepts from cases that might not be identified by straightforward similarity measures, and the less obvious the similarity the more creative the

solution will be. We are primarily interested in producing new concepts (the 'creative leap') rather than detailed instantiations.

Given a set of cases and an input problem our goal is to retrieve a concept contained in some exisiting case solution and use it to develop a new one. We are particularly interested in cases where the case problem is apparently dissimilar to the input problem. Our suggestion is that similarities may become apparent at higher levels of abstraction than is given in the original case and problem descriptions. Computational Intelligence (CI) techniques could be used to produce generalised concepts (that is, more abstract concepts that are incorporated in one or more less abstract cases). This process of generalisation could be repeated until a match is found and would apply to both case problems and solutions. The corresponding abstract solution concept would then form the basis of a new solution, which when instantiated would not be merely an adaptation of an existing solution at the same level of detail (Figure 2).

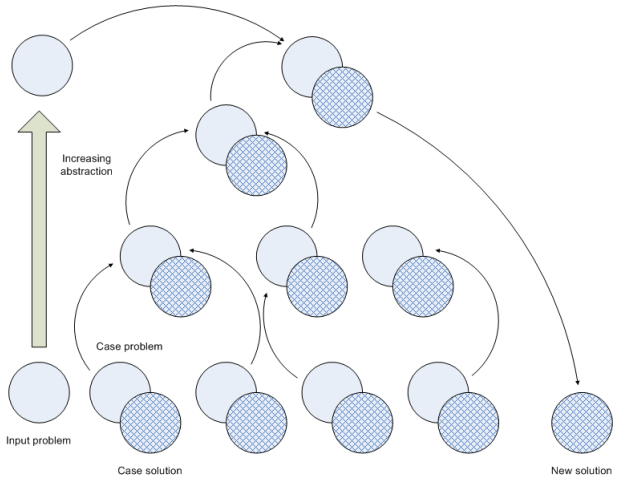


Fig. 2. Case-based creativity through generalisation

We suggest that CI techniques may support the operations and flexible generalisation needed to implement a Case-based Creativity (CBC) system as outlined above. CI techniques are essentially non-symbolic and include evolutionary computation and connectionist and fuzzy systems. These techniques have some useful properties and abilities such as tolerance to noise, clustering, generalisation, pattern matching, and (unsupervised) learning, and do not necessarily depend on their states mapping to valid examples in the input space, or the construction of a model of the space. The advantage of this approach is that we may be able to perform semantically meaningful generalisation without the overheads associated with reasoning over structured, symbolic data.

3.1 Graph representation

The model outlined above is very much symbolic in nature. This is unavoidable for working with problems which are in rich enough domains for creativity to have meaning.

Graphs provide a convenient way to represent structured, symbolic knowledge. Usually knowledge graphs are directed acyclic graphs such as semantic networks or concept nets. They are closely related to ontologies, which can be represented as graphs and provide both a vocabulary and a means of specifying relationships between entities (nodes in the graph). Graphs can also support induction and the definition of new relationships and concepts, useful for a system such as the one discussed. However doing this is computationally very expensive and we are not necessarily interested in the 'correct' formation of new concepts through induction; rather, we are looking for a mechanism to provide pointers to existing solutions whose relevance would not otherwise have been apparent. For this we need only know that two generalised concepts (i.e. higher level concepts produced through the abstraction of lower level ones) would be similar – whether these generalisations can be mapped to valid graphs with respect to some ontology is secondary.

In a case-based system individual cases might consist of a pair of graphs representing a solution and a problem.

3.2 Symbolic data and Computational Intelligence

An obvious issue is that CI techniques are sub-symbolic – that is, they work with real-valued vectors representing things in number space rather than symbol space. As mentioned above we need to use symbolic representations to work with meaningful design problems and creativity, and there is a 'semantic gap' between sub-symbolic and symbolic representations. However, some work has been done on using symbolic knowledge with CI [12] [13], typically involving recurrent neural networks or the recursive application of graphs to SOMs. Other approaches include some kind of preprocessing to reduce dimensionality, e.g. principal component analysis (PCA) or spectral analysis to identify the eigenvalues of a graph which can then be applied to a connectionist system.

If using CI techniques to generalise graphs corresponds to generalising the concepts that they represent, then the results will be semantically meaningful. Both Sparse Distributed Memory models (SDMs) [14] and Self-organising Maps (SOMs) [15] can generalise and retrieve items matching an input. In addition SOMs impose a topology on an input space, which may or may not be useful as we are primarily interested in which cases are similar to a given problem, not with each other. However, a SOM generalises over the input space on which it is trained. This means that weight vectors come to represent examples from that space that it has not necessarily seen but are representative of the topology of the input space. With reference to our high level model we might call this 'horizontal generalisation'. What we are interested in is 'vertical generalisation' to higher levels of abstraction, and how this might be achieved with approaches such as SOMs appears to be an open research question.

3.3 Concretising solutions

Rather than adapting an existing solution we will likely need to concretise an abstract solution. There is an argument to be made that identifying this solution constitutes the 'creative leap' and that the job of producing a more detailed solution or design is secondary and likely to be routine. Again, we only have our existing knowledge with which to work in order to turn an abstract concept into a more detailed solution. We may have the generalised concept we have chosen, the context-specific details from the input problem, specific cases incorporating the generalised solution, the generalised input problem, generalised case problems incorporating the concept, and maybe other cases selected 'non-creatively' (e.g. with a similarity measurement) if available.

4 Other implementation issues

4.1 Building a case base

The aim is to use a publicly available dataset with which to build a case base. This means developing a graph for a problem and another representation for a solution. There are several ontologies available in RDF format that provide the vocabulary and relationships necessary to describe concepts in the relevant domain, and these can easily be used to generate graphs.

There is likely to be a conflict between a rich enough, cross-domain dataset to allow for creativity and practical considerations (e.g. vocabulary size). A domain-specific dataset has the advantage of consistency and a relatively tightly controlled vocabulary, but everything is already in the same context so there is perhaps less scope for creativity.

However there still remains the issue of describing a problem. If possible we want to avoid doing this ourselves to keep the introduction of our own bias to a minimum, but even so it is likely that any publicly available dataset will need a significant amount of preprocessing.

One possibility is to use patent data. Patents are attractive because they cover all domains and are often described in terms of problems and solutions using very specific language. Another advantage is the possibility of comparing the output of the system with actual solutions.

Whichever domain/dataset is chosen it will ideally be human-meaningful; i.e. the outputs are such that humans can assess their creativity.

4.2 Scope

The scope of a system will depend on its intended role in the design process. We have argued that the essence of the creative leap is in the association rather than in the concretisation of the resulting concept. If we envisage the system serving to prompt human designers then highlighting the relevant solutions might be sufficient. However if we wish to automate the evaluation of the results we will need to construct a new case that can be compared to existing ones which,

depending on the generality of the retrieved solution, might involve some further generalisation and concretisation.

4.3 Evaluation

We will need to evaluate how well the mapping between concepts and their sub-symbolic representations is preserved during processing in order to evaluate any results for creativity. If we are confident that semantic generalisation is occurring then we may explore the concept of creative potential to evaluate the creativity of results.

5 Summary

We are suggesting that creative potential increases with abstraction, and that creativity is in fact meaningless at the lowest, sub-symbolic or instance level. This suggests that there may be a novel and interesting assessment of creativity based on the amount of processing done on the case base to produce a solution, or at which level of abstraction an isomorphism was found between the new problem and the abstracted case problem. This might be used in combination with graph distance metrics. However we will be interested in the combination of both the problem (context) and solution aspects of a case, which may involve more than one graph.

It is also important that any solutions do actually solve the problem at hand. The need to evaluate performance may limit the choice of domain, and implies that we may need to generate solutions at the sub-symbolic level in order to be able to model or simulate them for evaluation.

References

1. Welch, M.: Analyzing the tacit strategies of novice designers. Research in Science and Technological Education **17**(1) (1999) 19–34
2. Johnsey, R.: The design process- does it exist? International Journal of Technology and Design Education **5**(3) (1995) 199–217
3. Poincaré, H.: The Foundations of Science. Science Press, Lancaster PA (1913)
4. Schank, R.C., Cleary, C.: Making machines creative. In Smith, S., Ward, T.B., Finke, R.A., eds.: The Creative Cognition Approach. MIT Press (1995) 229–247
5. Koestler, A.: The Act of Creation. Picador, London (1964)
6. Gabora, L.: Cognitive mechanisms underlying the creative process. In: C&C '02: Proceedings of the 4th conference on Creativity & cognition, ACM (2002) 126–133
7. Wills, L., Kolodner, J.: Towards more creative case-based design systems. In: AAAI '94: Proceedings of the twelfth national conference on Artificial intelligence (vol. 1), American (1994) 50–55
8. Cunningham, P.: CBR: Strengths and weaknesses. In: IEA/AIE (Vol. 2). (1998) 517–524
9. Kolodner, J.: What is case-based reasoning? In Hofstadter, D., ed.: Case-based Reasoning. Morgan Kauffmann, San Mateo, CA (1993)

10. Smyth, B., Keane, M.T.: Experiments on adaptation-guided retrieval in case-based design. In: ICCBR. (1995) 313–324
11. Gero, J.S., Kannengiesser, U.: The situated function-behaviour-structure framework. Design Studies **25**(4) (2004) 373–391
12. Frasconi, P., Gori, M., Sperduti, A.: A general framework for adaptive processing of data structures. IEEE-NN **9**(5) (1998) 768
13. Hagenbuchner, M., Sperduti, A., Tsoi, A.C.: A self-organizing map for adaptive processing of structured data. Neural Networks, IEEE Transactions on **14**(3) (May 2003) 491–505
14. Kanerva, P.: Sparse Distributed Memory. Sparse Distributed Memory. MIT (1988)
15. Vesanto, J., Alhoniemi, E.: Clustering of the self-organizing map. IEEE-NN **11**(3) (2000) 586

Vignette-Based Story Planning: Creativity Through Exploration and Retrieval

Mark O. Riedl

School of Interactive Computing
Georgia Institute of Techology
85 Fifth Street NW, Atlanta, Georgia 30308, USA
riedl@cc.gatech.edu

Abstract. Storytelling is a pervasive part of our daily lives and culture that is being applied to computational systems for entertainment, education, and training. Because of the prevalence of story in non-interactive media such as books and movies, as well as interactive media such as computer games, automated generation of narrative content is an essential component of making computers creative. We describe artificial intelligence planning as a model of narrative creation and then detail how the retrieval and reuse of vignettes can facilitate the creation of narratives within the planning framework. Vignettes are fragments of story that exemplify certain familiar narrative situations. We show how this new approach to story generation is capable of exploring a larger space of creative solutions and can create more valuable stories.

Keywords. Story generation; exploratory creativity; planning; case-based reasoning

1 Introduction

Storytelling is a pervasive part of our daily lives and culture. Storytelling is particularly prominent in entertainment, where stories can be viewed as artifacts to be consumed by an audience. Story also plays a role in education and training, where stories and scenarios can be used to illustrate and guide. The production of these artifacts – stories and scenarios – is a primary activity in the entertainment industry and also a significant bottleneck in the educational and training industries. In an "on-demand" society, waiting for periodic updates to serial narratives – weekly television series, movie series, and novels – is not considered ideal. Likewise, players of computer games that rely on stories and quests can complete quests faster than design teams can create new quests. How do we handle the situation in which content consumption is scaled up to the point where content consumption outpaces content production? One way to overcome the bottleneck of content production is to instill in a computer system the creative ability to generate new content.

Because of the prevalence of story in non-interactive media such as books and movies, as well as interactive media such as computer games, we concern ourselves with the automated generation of narrative content. The issue is whether an automated story generation system can be considered creative enough or skilled

enough to be trusted to produce content – stories – that will be experienced by users. More generally, the output of a creative system, such as an automated story generation system, must be *novel*, *surprising*, and *valuable* [1]. Whether an artifact is valuable is subjective. For the purposes of this paper, we will consider the minimal requirements for a story artifact to be considered valuable if it (a) meets the intended purpose of its creation and (b) is sufficiently mimetic – appearing to resemble reality, but in a way that it is more aesthetically pleasing than reality. In brief, stories should be novel, but not so novel that they are unrecognizable [2].

Two well-recognized approaches to story generation are planning and "knowledge-intensive" [3] techniques such as case retrieval. In this paper we present a step towards automated story generation that combines planning and the retrieval of narrative fragment – which we call *vignettes* – that are known to be "good" examples of mimetic situations.

2 Related Work

Boden [1] distinguishes between creativity as exploration and creativity as transformation. Likewise, narrative generation systems can be categorized roughly with regard to whether they treat the problem of creating narrative content as a problem of search or the problem of adapting existing knowledge and cases to new contexts.

Search based narrative generation approaches include Tale-Spin [4], which uses a simulation-like approach, modeling the goals of story world characters and applying inference to determine what characters should do. Dehn [5] argues that a story generation system should satisfy the goals of the human user. That is, what outcome does the user want to see? The Universe system [6] uses means-ends planning to generate an episode of a story that achieves a user's desired outcome for the episode. More recent work on narrative generation attempts to balance between character goals and human user goals [7; 8]. Further work on story planning addresses expanding the space of stories that could be searched [9].

Treating the problem of narrative generation as adapting existing knowledge has lead to variety of approaches that use case-based reasoning and/or analogy. Minstrel [10] implements a model of cognitive creativity based on routines for transforming old stories into new stories in new domains. ProtoPropp [3] uses case-based reasoning to generate novel folk tales from an ontological case base of existing Proppian stories. Mexica [11] uses elements of both previous story retrieval and means-ends planning.

Case-based reasoning (c.f. [12]) has been found to be related to creativity [1; 13]. The story planning algorithm that we describe in Section 3 is reminiscent of a certain class of case-base reasoners called transformational multi-reuse planners. Transformational multi-reuse planners are planners that attempt to achieve given goals by identifying and retrieving solutions to similar, previously solved problems. A multi-reuse planner may attempt to combine several cases in order to solve the given problem. Our story planning algorithm is similar in many ways to [14] and [15]. We compare and contrast our algorithm to these at the end of Section 3.2.

3 A Computational Approach to Generating Stories

We view story generation as a problem-solving activity where the problem is to create an artifact – a narrative – that achieves particular desired effects on an audience. We favor a general approach where we model the story generation process as planning (c.f. [6], [7], [8], [9]). Conceptually a planner can be thought as an approach to problem solving in which *critics* [16] are applied to incomplete solutions until a complete solution is found. Each critic inspects a plan for a different type of flaw. We conceive of a critic as containing two parts: a flaw recognizer and a flaw repairer. The flaw recognizer component of each critic is invoked after any new potential solution plan is generated. If a critic recognizes a flaw in the plan, it annotates the plan to indicate that the plan cannot be considered a valid solution because of a flaw. The planner chooses an incomplete plan to work on and chooses a flaw to work on. The appropriate critic's flaw repairer routine is invoked, which proposes zero or more new plans in which the flaw is repaired (and often introducing new flaws). For example, one type of critic recognizes and repairs *open condition flaws*. An open condition flaw exists when an action (or the goal state) in a plan has a precondition that is not established by a preceding action (or the initial state). The critic can repair this flaw by applying one of the following repair strategies:

(i) Selecting an existing action in the plan that has an effect that unifies with the precondition in question.

(ii) Selecting and instantiating an operator from the domain operator library that has an effect that unifies with the precondition in question.

The planner uses special annotations called *causal links* to indicate when open conditions are satisfied. A causal link establishes the causal relationship between two actions in the case that the effects of the former action establish a condition in the world necessary for the latter action to execute successfully. The set of critics used in conventional planners such as [17] assure plan that plans are sound, meaning that they are guaranteed to execute successfully in the absence of unanticipated changes in the world. However, stories are much more than just ways of achieving an intended outcome in the most efficient manner. Stories should meet the expectations of the audience. This may mean putting in details that are aesthetically pleasing even if they are not strictly necessary.

When humans write stories, they call on their lifetime of experiences as a member of culture and society. A computer system that generates stories does not have access to this wealth of information. As a way of mitigating this handicap, a computer system can be provided with a wealth of knowledge in the form of traces of previous problem-solving activities or libraries of previous solutions (e.g. stories). One "knowledge intensive" approach [3] is to use a form of case-based reasoning. Our story planning algorithm achieves longer and more mimetic narrative sequences by accessing a library of vignettes – fragments of stories that capture some particular context. We do not presume to know how these vignettes were created, only that we have the solutions and that they have favorable mimetic qualities.

```
Vignette:
Steps: 1: Start-Battle (?c1 ?c2 ?place)
       2: Wound (?c1 ?c2)
       3: Wound (?c1 ?c2)
       4: Mortally-Wound (?c2 ?c1)
       5: Die (?c1)
       6: End-Battle (?c1, ?c2)
Constraints: (character ?c1)
             (character ?c2)
             (stronger ?c2 ?c1)
Ordering: 1→2, 2→3, 3→4, 4→5, 4→6
Causation: 1→(battling ?c1 ?c2)→2
           1→(battling ?c1 ?c2)→3
           1→(battling ?c1 ?c2)→4
           1→(battling ?c1 ?c2)→6
           4→(mortally-wounded ?c1)→5
Variable-constraints: ?c1 ≠ ?c2
Effects: (battling ?c1 ?c2)
         (not (battling ?c1 ?c2))
         (wounded ?c2)
         (mortally-wounded ?c1)
         (not (alive ?c1))
```

Fig. 1. An example vignette data structure describing a battle in which a weaker character (*?c1*) takes on a stronger character (*?c2*) and dies.

3.1 Vignettes

We use the term *vignette* to refer to a fragment of a story that represents a "good" example of a situation and/or context that commonly occurs in stories [18]. For example, a library of vignettes would contain one or more specific instances of bank robberies, betrayals, cons, combat situations, etc. It is important to note that the library contains specific examples of these situations instead of general templates. The implication of the existence of this library is that a story generator does not need to "reinvent the wheel" and thus does not need the specialized knowledge required to be able to create specialized narrative situations. Vignettes are fragments of story structure. Unlike a script, a vignette can be thought of as a fragment of a narrative – a partially ordered set of events that are perceived to be essential in presenting a narrative situation. How does one know what actions should be included in the vignette and which can be left out? We use the minimal vignette rubric: a *minimal vignette* is one in which removing any one action from the vignette causes it to no longer be considered a good example of the situation and/or context it was meant to represent.

Computationally, vignettes are stored as plan fragments. As a plan fragment, it is possible that some actions do not have to have all of its preconditions satisfied. This is a way of saying that it is not important how the situation is established or even why, but once the conditions are established certain things should happen. Vignette plan fragments do not reference specific characters, objects, or entities so that a planner can fit the vignette into new story contexts by making appropriate assignments. To ensure illegal or non-sense assignments are not made, co-designation and non-co-designation variable constraints are maintained. Fig. 1 shows an example vignette capturing a very simple combat between two characters where

one character (represented by the variable *?c2*) is stronger than the other (represented by the variable *?c1*). The weaker character wounds the stronger character twice before the stronger character delivers a mortally wounding blow. Finally, the mortally wounded character dies of its wounds. This vignette could be used in any plan in which a character must become wounded, mortally wounded, or dead.

3.2 Planning with Vignettes

The Vignette-Based Partial Order Causal Link (VB-POCL) planner is a modification of standard partial order planners to take advantage of the existence of a knowledge base of vignettes. The VB-POCL planning algorithm is similar to other multi-reuse case-based planners such as [14] and [15] in that it modifies the open condition critic by adding a third strategy for repairing open condition flaws:

(iii) Retrieve and reuse a case that has an action with an effect that unifies with the precondition in question.

Given an action in the plan that has an unsatisfied precondition VB-POCL non-deterministically chooses one of the three above strategies. Strategies (i) and (ii) are performed in the standard way (c.f., [17]). If strategy (iii) is selected, VB-POCL's open condition critic retrieves a vignette that has an action with an effect that will satisfy the unsatisfied precondition. The retrieval process is one of identifying a vignette in the knowledge base that has an action that has an effect that unifies with the open precondition that the open precondition critic is trying to satisfy. But the critic does not splice the vignette into the flawed plan. Instead, another critic recognizes that the plan if flawed because the vignette has not been fitted into the plan and annotates the plan with a new type of flaw called a *fit flaw*. A fit flaw is repaired only when all the actions in the retrieved vignette have been instantiated in the plan. Repairing a fit flaw is a process of selecting an action from the retrieved vignette and adding it to the new plan (or selecting an existing action in the plan that is identical to the selected action to avoid unnecessary action repetition) with all relevant causal links, temporal links, and variable bindings.

It may take several invocations of the fitting critic to completely repair a fit flaw. This may seem more inefficient than just adding all vignette actions to the plan at once. There are three advantages to iterative fitting. First, it is easier to recognize and avoid action repetition. Second, it allows other critics to observe the plan at intermediate stages of fitting in case there are interesting synergies between critics. For example, fitting may lead to the creation of new open condition flaws that in turn are repaired through conventional planning (strategies i and ii) or by retrieving new vignettes (strategy iii). Third, problems in the fitting process can be identified sooner in case the strategy must be abandoned. One of the interesting properties of VB-POCL – shared with other case-based planning algorithms – is that it can operate when there are no applicable vignettes available; the algorithm can fall back on conventional planning. If applicable vignettes are available, plan-space search control heuristics are required to prevent a potential explosion of open conditions.

VB-POCL is a variation on one type of case-based reasoning called *transformational multi-reuse planning* (c.f. [14], [15]). Transformational multi-reuse planners attempt to reuse components of solutions to similar problems to solve new

problems. VB-POCL is a variation on transformational multi-reuse planning; vignettes are not solutions to previous problems and VB-POCL does not attempt to learn to solve problems from past examples. That is, VB-POCL does not *retain* its solutions because they are neither vetted nor minimal, as vignettes are required to be. VB-POCL relies on certain assumptions that make it different from other case-based reasoning techniques. First VB-POCL assumes that vignettes are minimal. Because of this, VB-POCL doesn't stop fitting a vignette until all actions in the vignette are present in the new plan, even if some actions do not serve a causal purpose. This is in contrast to other case-based reasoning techniques that discard actions that are strictly unnecessary from the perspective of achieving a goal state. Second, VB-POCL assumes that vignettes in the library are in the domain of the story being generated. The implication of this assumption is that the planner does not need to deliberate about the cost tradeoff between using standard planning versus retrieval and reuse (which is otherwise very high). VB-POCL non-deterministically chooses between flaw repair strategies (i and ii) and (iii), meaning that it applies all strategies to each and every flaw by branching the search space and exploring each branch in turn. This is not practical if vignettes require extensive modifications for reuse. To support this property of VB-POCL, we use an offline algorithm based on analogical reasoning to pre-process the vignette knowledge base and transform vignettes from their native story world domains to the domain of the new story to be generated [18]. The vignette transformation process is overviewed in Section 3.4.

3.3 Example

To illustrate the VB-POCL planning algorithm, we provide an example of how the planner could use the vignette shown in Fig. 1. Suppose we wanted a story set in J.R.R. Tolkein's Middle Earth. The story world is in the state in which one character, called *Enemy*, has in his possession a Silmiril – a precious magical stone. The outcome, provided by the human user, is that another character, called *Hero*, gains possession of the Silmiril. The planner starts by non-deterministically choosing to satisfy the goal by having Hero take the Silmiril from Enemy. This requires that the Enemy not be alive. The planner could use the vignette from Fig. 1 here by retrieving it and binding Enemy to *?c1*. Note that this strategy will eventually fail because it would require a character stronger than Enemy. Instead the planner solves the problem chooses to instantiate an action, *Die-of-Infection*, that causes Enemy to not be alive. This requires that Enemy be superficially wounded. Here VB-POCL retrieves the vignette from Fig. 1 because it has an action that can have the effect (once variables are bound) of causing Enemy to become wounded.

Each vignette action is spliced into the new story plan one at a time. The temporal and causal relationships between vignette actions are maintained in the new plan. However, any time a new action is added to the plan, causal threats may arise in which the new action potentially undoes an existing causal relationship. These causal threats are handled by imposing additional temporal constraints on actions in the plan (or if the threat cannot be resolved, the planner backtracks) [17]. For example, when *Die(Hero)* is spliced into the story plan, it must be temporally ordered after *Take* to

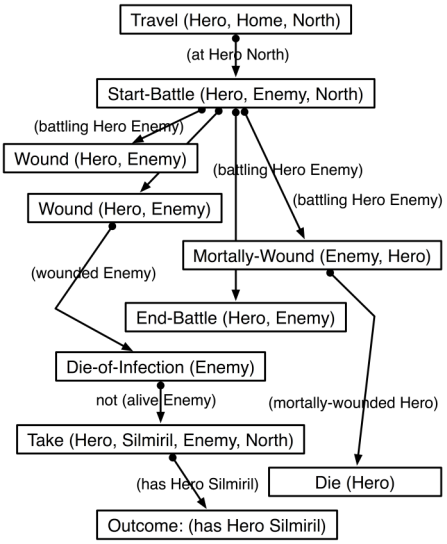


Fig. 2. An example story plan generated by VB-POCL. Boxes represent actions or events and arrows represent causal relationships between those actions.

avoid inconsistencies; a dead characters cannot perform actions. The order that actions are chosen from the vignette and spliced into the plan does not matter.

When a vignette is retrieved, one action is chosen non-deterministically to be the *satisfier action*. This is the action that will be used to satisfy the original open condition flaw that triggered the case retrieval in the first place. In the example, the satisfier action is one of the *Wound* actions because it has the effect of causing the Enemy to become wounded. When the satisfier action is spliced into the plan, the planner takes the extra step of causally linking the satisfier to the action with the original unsatisfied precondition – in this case, Enemy needing to be superficially wounded in order to die of an infection – that triggered the vignette's retrieval in the first place. Thus the open condition flaw is finally repaired.

The vignette is fairly self-contained, but the vignette action, *Start-Battle* does require that the planner establish that both Hero and Enemy are at the same place, which in this case the North. This precondition is satisfied in the normal way, by instantiating an action in which Hero travels to the North (Enemy is already there). The final story plan is shown in Fig. 2. Boxes are actions and arrows represent causal links. A causal link indicates how an effect of one step establishes a world state condition necessary for a precondition of latter steps to be met. For clarity, only some preconditions and causal links on each action are shown.

3.4 Vignette Transformation

VB-POCL assumes a library of vignettes that are already in the domain of the story to be generated. A domain is a set of propositions that describe the world, including

characters, and a set of operator templates that described what characters can do and ways in which the world can be changed. In the example, the domain describes characters such as Hero and Enemy and operators such as *Travel* and *Wound*. However, we may want the story planner to have access to vignettes from other domains, especially if our new story is set in an unique and specialized story world domain. To transfer vignettes between domains, one must first find analogies between domains. This differs from the problem of finding analogies between stories because there is not a second instance of a story to compare. Instead, we search for analogies between domains and use that information to translate a known vignette from one domain to another. The transfer process is summarized as follows. A *source vignette* is a vignette in an arbitrary domain, called the *source domain*. The *target domain* is the domain of the story to be generated. For each action in the source vignette, the far transfer algorithm searches for an action in the target domain that is most analogical. The search involves a single-elimination tournament where target domain actions compete to be the most analogical according to the Connectionist Analogy Builder (CAB) [19]. The winner of the tournament is the target domain action most analogical to the source domain action. The result is a mapping of source domain actions to target domain actions that can be used to translate a source vignette into a target domain through substitution. Translated vignettes may have gaps where translation is not perfect. This is not a problem because the VB-POCL will recognize this and fill in the gaps via planning. Applying this process to all vignettes in a library results in a new library in which all vignettes are in the proper domain. See [18] for a more detailed description of the algorithm.

4 Discussion

One of the interesting properties of VB-POCL is that vignette retrieval can result in story plans in which there are actions that are not causally relevant to the outcome. Trabasso [20] refers to actions that are causally irrelevant to the outcome as dead-ends. In the example above, the causal chain involving Enemy mortally wounding Hero and then Hero dying appears to be a dead-end because those actions do not contribute to Hero acquiring the Silmiril. Psychological studies indicate that dead-ends are not remembered as well as actions that are causally relevant to the outcome [20], suggesting that dead-ends should be avoided. For example, a battle in which a single wound was inflicted on Enemy would have sufficed, and this is what planners such as [7; 8] and [17] would have settled on. However, human authors regularly include dead-end events in stories suggesting some importance to dead-ends. We hypothesize that there are certain mimetic requirements to be met in any story and that dead-ends can serve this purpose. For example, we assume that a combat scenario in which many blows of varying strengths are exchanged is more interesting than a combat in which a single blow is dealt. Interestingly, what may be a dead-end causal chain to the story planner may not be considered a dead-end by a human reader, and vice versa. That is, the reader may interpret the example story as a *tragedy* and consider the death of Hero as one of *two* primary causal chains, whereas the planner's representation contains only *one* causal chain that leads to the human

user's imposed outcome (Hero has the Silmiril). More research needs to be done to create intelligent heuristics to recognize when dead-ends (from the planner's perspective) are favorable, tolerable, or damaging.

Does VB-POCL improve the computational creativity of planning-based story generation approaches? Planning-based story generators treat creativity as exploratory search [9], which is one of two perspectives on creativity offered by Boden [1]. VB-POCL brings exploratory and transformational creativity closer together. Incorporating case-based retrieval into a planning framework suggests that transformation is a special instance of exploration. Conceptually, we can consider plot generation as a search through the space of all possible narratives, where a narrative is a set of temporally arranged events that change the story world (for the purposes of completeness we also consider the empty story). In the space of all possible narratives, narratives that are adjacent differ in just one detail – an extra event or a different temporal ordering of events. One can walk the space of all possible narratives, beginning with the empty narrative by applying the operator, *add-event(e, c)*. This operator moves one from a narrative to an adjacent narrative that differs in that it contains one additional event, *e*. The parameter *c* is a set of constraints that unambiguously positions *e* temporally relative to other events in the narrative. The operator *add-event* is an abstraction of the iterative process used by planners in the search for the first visited structure to which critics do not attribute flaws.

Because of the iterative nature of vignette fitting in VB-POCL, a vignette can be viewed as a strategy for guiding the walking of the space towards a sub-space of stories that are more valuable. Because vignettes are made retrievable by transforming them via analogical reasoning, we hypothesize that transformational creativity may be a special case of exploratory creativity. That is, transformation processes *short-circuit* exploration by using knowledge from seemingly unrelated domains to specify a target or direction for search. In the case of VB-POCL, however, one could claim that some or all of the creativity occurs prior to exploration in the offline process that transformed vignettes into the correct – applicable – domain via analogical reasoning.

These vignette-guided walks also extend beyond the boundaries of the space that can be walked by conventional planning techniques – specifically visiting stories in which there are actions that are not causally necessary. As noted in [9], expanding the space that can be explored provides an opportunity to find more solutions that are valuable. We believe that VB-POCL searches a space of stories that has more valuable and mimetic solutions. This is partially achieved by attempting to retrieve vignettes whenever possible. Future work involves identifying heuristics that can determine when retrieving a particular vignette is more likely to lead to a valuable solution. Future work also involves incorporating additional properties of character and story into the algorithm such as character intentionality [7; 8], character personality [9], and emotion [11].

We find planning to be a valuable model for story generation, in general. One reason for this is that plans are reasonable models of narrative [21]. But also planners walk the space of possible narratives in search of a solution that meets certain qualities. VB-POCL extends the general planning algorithm by retrieving and reusing vignettes. This is a strategy for tapping into the experiences of other presumably expert story authors. Interestingly, VB-POCL can explore a greater space

of stories because it can consider story plans that have action that are not causally necessary to reach some given outcome. We believe that some of these stories will be more valuable because of the mimetic qualities of the vignettes and the potential for these stories to possess both global novelty and localized familiarity.

References

1. Boden, M.: *The Creative Mind: Myths and Mechanisms, 2nd Edition*. Routledge, New York (2004).
2. Giora, R.: *On Our Mind: Salience, Context, and Figurative Language*. Oxford University Press (2003).
3. Gervás, P., Díaz-Agudo, B., Peinado, F., Hervás, R. Story Plot Generation Based on CBR. *Journal of Knowledge-Based Systems*, 18(4-5), 235-242 (2006).
4. Meehan, J.: *The Metanovel: Writing Stories by Computer*. Ph.D. Dissertation, Yale University (1976).
5. Dehn, N.: Story Generation after Tale-Spin. In: *7th International Joint Conference on Artificial Intelligence* (1981).
6. Lebowitz, M.: Story-Telling as Planning and Learning. *Poetics*, 14, 483—502 (1985).
7. Riedl, M.O. *Story Generation: Balancing Plot and Character*. Ph.D. Dissertation, North Carolina State University (2004).
8. Riedl, M.O., Young, R.M.: An Intent-Driven Planner for Multi-Agent Story Generation. In: *3rd International Joint Conference on Autonomous Agents and Multi Agent Systems* (2004).
9. Riedl, M.O., Young, R.M.: Story Planning as Exploratory Creativity: Techniques for Expanding the Narrative Search Space. *New Generation Computing*, 24(3), 303—323 (2006).
10. Turner, S.: *MINSTREL: A Computer Model of Creativity and Storytelling*. Ph.D. dissertation, University of California , Los Angeles (1992).
11. Pérez y Pérez, R. and Sharples, M.: Mexica: A Computer Model of a Cognitive Account of Creative Writing. *Journal of Experimental and Theoretical Artificial Intelligence*, 13(2), 119—139 (2001).
12. Kolodner, J.: *Case-Based Reasoning*. Morgan Kaufmann Publishers (1993).
13. Kolodner, J.: Understanding Creativity: A Case-Based Approach. In: *1st European Workshop on Case-Based Reasoning* (1993).
14. Francis, A.G., Ram, A.: A Domain-Independent Algorithm for Multi-Plan Adaptation and Merging in Least-Commitment Planners. In: *AAAI Fall Symposium on Adaptation of Knowledge for Reuse* (1995).
15. Britanik, J., Marefat, M.: CBPOP: A Domain-Independent Multi-Case Reuse Planner. *Computational Intelligence*, 20, (2004).
16. Sacerdoti, E.D.: A Structure for Plans and Behavior. Elsevier, New York (1977).
17. Weld, D.: An Introduction to Least Commitment Planning. *AI Magazine*, 15(4), 27-61 (1994).
18. Riedl, M.O., León, C.: Toward Vignette-Based Story Generation for Drama Management Systems. In: *2nd International Conference on Intelligent Technologies for Interactive Entertainment, Workshop on Integrating Technologies for Interactive Story* (2008).
19. Larkey, L.B. and Love, B.C.: CAB: Connectionist Analogy Builder. *Cognitive Science,* 27, 781—794 (2003).
20. Trabasso, T., van den Broek, P.: Causal Thinking and the Representation of Narrative Events. *Journal of Memory and Language*, 24, 612-630 (1985).
21. Young, R.M.: Notes on the Use of Plan Structures in the Creation of Interactive Plot. In: *AAAI Fall Symposium on Narrative Intelligence* (1999).

Creative Storytelling Based on Exploration and Transformation of Constraint Rules*

Carlos León[1] and Pablo Gervás[2]

Departamento de Ingeniería del Software e Inteligencia Artificial
Universidad Complutense de Madrid
[1]cleon@fdi.ucm.es, [2]pgervas@sip.ucm.es

Abstract. It is said in Boden's work that transformational creativity is the highest form of creativity. Whether this claim is true or not can be discussed, but, in principle, *transforming* the conceptual space should lead creativity processes to new areas not covered simply by exploratory creativity. In this paper Cast is presented, a new automatic storytelling system which has the ability of transforming this conceptual space by performing a state space search generating facts beyond the expected set of valid solutions and modifying the constraints for the generative process. Why this system can be considered creative is discussed, and example results from the first prototype are shown.

Keywords: Computational Creativity, Transformational Creativity, Cast, Storytelling.

1 Introduction

Boden's work considers two types of creativity regarding *conceptual space*, (the abstract location of the entities produced by creative processes): *exploratory* (exploring or searching the conceptual space) and *transformational* (changing or transforming the conceptual space) [1,2]. It is said that transformational creativity is the "highest form", that is, for a computer system to be fully creative, it has to *transform* the conceptual space.

There is not much consensus about these ideas: Bundy, in [3], explains that Boden's model does not cover all possible creative productions, and Ritchie, in [4], trying to obtain a clear definition of transformational creativity, concludes that terms used to explain what it is are still too imprecisely defined to claim that a process or an artifact has been creative in terms of high level of creativity. However, many researchers in the field agree that, even without precisely, empirically knowing the definition of creativity in general and transformational creativity in particular does not prevent us from building creative systems: those

* This research is funded by the Ministerio de Educación y Ciencia (TIN2006-14433-C02-01), Universidad Complutense de Madrid, Dirección General de Universidades e Investigación de la Comunidad de Madrid (CCG07-UCM/TIC 2803) and a grant from Caja de Burgos, Jóvenes Excelentes 2008 contest.

that *would be deemed as creative if found in humans* (as claimed, for example, by Wiggins in [5]).

Against this background, and related with Computational Creativity, automatic story generation programs follow a long tradition of creative systems, like TALE-SPIN, MINSTREL or MEXICA[6,7,8]. Typically, these systems are built with exploratory creativity processes. In order to explore new approaches to applied Computational Creativity in general and automatic story telling in particular, a story teller called CAST (*Creative Automatic Story Teller*), which tries to modify the conceptual space, is presented. Transformational creativity processes which are implemented by CAST, although not totally directed, are heavily influenced by theoretical models by Sharples ([9]), and also in previous story telling systems like MEXICA [8], KIIDS [10] and FABULIST [11].

Even considering Boden's account for creativity, nothing guarantees, in all cases, that the final quality of the generated stories will be higher than quality obtained by applying just exploratory creativity. Creative systems can be extremely complex, and we have not defined measurements for aesthetics in evaluating creative stories. However, expanding the conceptual space by modifying it opens new possibilities for generation, so possibly a new creative object (a new story, in this case), following rules not previously conceived by the system creator, could be generated. CAST uses both transformational and exploratory processes, so it should benefit from both approaches. This system does not claim to create better stories just by applying transformation. The present paper shows a way of changing the search space which is implemented in CAST, showing that it is possible to transform the constraints to get new information. Whether this is what Boden defines as transformational creativity or not is arguable, given that the definition is far from being precise enough. CAST just tries to show that transformation is *possible*, and *useful*. We will see, in the next sections, how this is done and the preliminary results that have been obtained.

2 Previous Work

Storytelling is crucial to understand the way humans perceive the world. The task of conceiving and composing the story that is to be told is a fundamental ingredient of storytelling. There are several approaches to automatic storytelling. The earliest attempt at addressing explicitly issues of creative activity in the context of storytelling is the development of MINSTREL [7], which tells stories about King Arthur and his Knights of the Round Table . BRUTUS [12] is a program that writes short stories about betrayal. It has a complex architecture designed to take into account knowledge about literature, but it addresses very few of the issues raised around creativity either from the point of view of writing as a cognitive activity or from the point of view of computation. MEXICA [8] implements a computer model designed to study the creative process in writing in terms of the cycle of engagement and reflection. MEXICA is a flexible tool where the user can set the value of different parameters to constrain the writing process and explore different aspects of story generation. It takes into account emotional

links and tensions between the characters as means for driving and evaluating ongoing stories. MEXICA is actually built based on Sharples' account of writing as a cycle of engagement and reflection phases. However, not all storytelling or narrative/linguistic related systems are directly based on the study of creativity. Works like FABULIST [11], TALE-SPIN [6] or UNIVERSE [13] in the narrative field, or [14] in poetry are some other examples of storytelling which follow other narrative models.

Wiggins [5] takes up Boden's idea of creativity as search over conceptual spaces and presents a more detailed theoretical framework that attempts to clarify the issue of whether creativity can be reduced to good old fashioned AI search (GOFAI). By specifying formally the different elements involved (the universe of possible concepts, the rules that define a particular subset of that universe as a conceptual space, the rules for traversing that conceptual space) Wiggins points out that the rules for traversing a conceptual space may lead to elements in the universe but outside the definition of the conceptual space. This kind of behavior would be considered a design error in an AI system, but one must keep in mind that his formal framework is intended as a characterization of the various elements that may be at play in a creative situation.

Sharples [9] presents a description of writing understood as a problem-solving process where the writer is both a creative thinker and a designer of text. Sharples proposes a cyclic process moving through two different phases: engagement and reflection. During the engagement phase the constraints are taken as given and the conceptual space defined by them is simply explored, progressively generating new material. During the reflection phase, the generated material is revised and constraints may be transformed as a result of this revision. Sharples also define two concepts which are relevant for this article: the *universe*, which is a set containing all possible stories, and the *conceptual space*, which is the result of restricting the set of coherent or valid stories. This *conceptual space* can be obtained by applying known *schemas*, or know rules about a given domain (for instance, knowing what is the process for buying a newspaper).

3 Cast: Storytelling Based on Exploration and Transformation

In this section, CAST is presented. The explanation is divided in three parts: The first one explains how the knowledge of the system is represented and managed (Section 3.1), the second one explains how the exploration is performed and what can be obtained by plain exploration in the toy domain that has been created (Section 3.2), and the third one shows how transforming constraints for the generation can lead the system to new results, not obtainable by just exploration (Section 3.3).

3.1 Representing Knowledge: Schemas and Facts

Basic knowledge in CAST is defined as a big set of logic predicates, in which the name of the predicate is a property which links two objects, in the form

$chase(policeman, criminal), want(criminal, money)$. It is also possible to define these facts with instantiation of logic variables: Instead of defining each possible group of $property(object, value)$, we can define facts with logic rules. For a very simplistic instance: $chase(?x, ?y) \leftarrow works(?x, policestation) \wedge steals(?y, money)$.

Constraints in CAST are stored in pieces of information called *schemas*. Schemas in CAST are sets of rules which define particular knowledge about something in the domain. For instance, to create a constraint which sets that a policeman can not be a friend of a criminal, we could write a small logic constraint, as a schema, like $friend(?x, ?y) \wedge \neg chase(?x, ?y)$. The symbols with a ? mark are logic variables, and can be bound to literals like *policeman* or *criminal*.

3.2 Creating the Story by Exploring the Conceptual Space

To explore the conceptual space, CAST performs a state space search in which a partial story is completed by the addition of new facts during the exploration. We have a knowledge base $\langle K, C \rangle$, where K is the set of logic facts, and C the set of schemas (in the form explained in Section 3.1), and a set with user restrictions in the form of logic rules for the story generated which we call I. Both sets have to be created by external systems, by hand or by analyzing texts, for instance. In these example they have been created by hand for explanation purposes. K stores a set of logic facts about a toy domain where there is a policeman, a criminal and a warehouse, and constraints in C are basic contraints about the world. We also have a partial initial story s_0, which is the empty set of facts.

Each new state in the search adds a subset σ_K of K, which will contain several facts. To choose this subset, we have to define a function $\varphi(s, K)$ which will return the new subset to be added to s, the partial story. There are virtually infinite possible definitions of $\varphi(s_i, K)$, so, for the sake of simplicity, we define it as a function which returns a random set of facts from K which were not present in s_i. Obviously, this definition is far from being the best possible one, but it will serve for our purpose of showing how transformation could improve the generation of a new story. In Section 4 how choosing a best $\varphi(s_i, K)$ could improve the story generation is discussed. The system, on each step, will apply the next operator:

$$s_{n+1} = s_n \cup \varphi(s_n, K) \tag{1}$$

Then the search will start generating different partial stories. On each step (each partial story s_i), before expanding the children nodes, the system has to determine if the partial story is *coherent* as a story. Intuitively, this function would return that $kill(criminal, policeman), eat(policeman, sandwich)$ is not coherent, because (considering these facts to be sequential) a dead man can not eat anything. To define this function, $\lambda(s_i, C)$, we use the set of schemas C. $\lambda(s_i, C)$ will return *true* if each fact of the set of the partial story $s_i$is valid with the schema base. For instance, if $C = \{friend(?x, ?y) \wedge \neg chase(?x, ?y)\}$, and $s_i = \{chase(policeman, criminal) \wedge friend(policeman, criminal)\}$, $\lambda(s_i)$ will return $false$.

Algorithm 1 Exploratory algorithm for Story Generation

1. $s_0 = \{\}$
2. while $\omega(s_i, I) = \bot$/* *while the partial story is not coherent with the user* */
   (a) repeat $s_{n+1} = s_n \cup \varphi(s_n, K)$ while $\lambda(s_{n+1}, C) = \bot$/* *look for a new fact* */
   (b) if it has been possible, *continue*, expand the new state
   (c) if there is no possible fact, return *failure*, discard the fact and continue exploring

$$\lambda(s_i, C) = \begin{cases} \bot \leftarrow \exists c \in C \mid c \wedge s_i \rightarrow \bot \\ \top \leftarrow otherwise \end{cases} \tag{2}$$

For each s_i, in each state of the search, we have another function which decides whether the new partial story s_i is valid as a story as the user wants it to be, that is, what kind of story the user wants. If it is valid, then it is returned as a solution. This function must be controlled by the user requirements, as the constraints set for the story, what we call I. The function is called $\omega(s_i, I)$. For instance, to set that the story should be about a policeman which captures a criminal. Formally, it could be expressed as a set of facts which have to be true in the story, the definition is the next one:

$$\omega(s_i, I) = \begin{cases} \top \leftarrow I \subset s_i \\ \bot \leftarrow otherwise \end{cases} \tag{3}$$

All this description of the exploration process just defines a generative search where we check the validity of the partial story with two functions: One of them is defined in the knowledge base, and the other one is defined by the user. Obviously, with this description, we can generate a story which will be a subset of K.

The algorithm, then, could follow a simple state space search. Algorithm 1 gives a global idea of how the process is performed.

So, having defined $\langle K, C \rangle$ and I and the functions $\varphi(s_i, K)$, $\lambda(s_i, C)$ and $\omega(s_i, I)$, we could obtain this story about a criminal who steals something, and a policeman chasing him (we assume that the order of the facts is the sequence of them in time, first the left column from top to bottom, and next the right one):

$$\begin{array}{cc} getin(criminal, bank) & shoot(criminal, police) \\ catch(criminal, money) & shoot(police, criminal) \\ go(criminal, warehouse) & kill(policeman, criminal) \\ chase(policeman, criminal) & feel(policeman, good) \end{array}$$

This story is simple, but coherent. However, nothing special has been performed to create it, and it is not creative. It is just a set of valid facts together, and all the information needed for this story to be created has been set by the

designed of the system and the functions. Although defining the previous functions in a different, more sophisticated way, could create better stories (in the sense of more complex, or more detailed), the process would not be creative (although the functions could be computed with creative processes).

3.3 Creating the Story by Transforming the Constraints in the Conceptual Space

In the previous section we have seen that exploring the conceptual space can lead to coherent results, but the quality is limited. Of course, there are better ways of exploring the conceptual space. What we present in this section is how it is possible to adapt the state space search for it to be able to modify the constraints of the $\lambda(s_i, C)$ function.

We have seen that for each modification of the partial story, a set of new facts is chosen, and then, if the facts are *coherent*, a new partial story is generated, and then the search for a valid story continues. CAST creates many non-coherent partial stories, and, just by exploratory processes, they are simply discarded. These non-coherent facts *break* the conceptual space, but that does not necessarily mean that they lead to a non-valid story.

Thus, a new option arises for the process: Adapting the set of constraints to let the conceptual space grow, and in this way letting the system create a set of stories, which is different that the set which plain exploratory methods like the previously explained one can create.

In order to make this ideas possible in CAST, the previous algorithm has to be modified: The C set will change during the generation process. Then, we have to replace the $\lambda(s_i, C) : \langle set, set\rangle \rightarrow boolean$ function, with:

$$\lambda'(s_i, C_i) : \langle set, set\rangle \rightarrow \langle boolean, set\rangle$$

which receives the state from the state space search, and returns *true* if the new s_i (the new partial story) can be made coherent, and a new set of constraints, which is a transformation of the previous one. Of course, the transformation of the constraint rules could be done even if the new state is coherent, but the process would be the same.

The definition of a new λ' function involves creating a model of how transversing the constraints creates acceptable stories. While creating such a model and precisely defining what an "acceptable" story is, are beyond the scope of this paper, it is interesting to give some ideas which could be useful for discussion (Section 3.3).

We could define λ' as a function which just considers a percentage of not-valid facts (as defined in Section 3.2) as valid ones, using a random function. For instance, if we allow to add 10% of non-valid facts. This is semantically equivalent to the next expression (j is the state which follows the state i):

$$\lambda'(s_j, C_j) = \begin{cases} \bot, C_i \leftarrow (\exists c \in C \mid c \wedge s_i \rightarrow \bot) \wedge random(0..1) \geqslant param \\ \top, C_i \leftarrow otherwise \end{cases} \quad (4)$$

This approach can be considered as noise in the system, and can lead to unexpected results. Some of the can be considered coherent, and somehow original. At least, not possible by plain exploration. In this story (chosen in a set of generated stories with these process, many of them not coherent), a corrupt policeman betrays a criminal:

$$\begin{array}{cc} getin(criminal, bank) & getin(policeman, car) \\ catch(criminal, money) & go(policeman, warehouse) \\ friend(criminal, policeman) & go(criminal, warehouse) \\ getin(criminal, car) & kill(policeman, criminal) \end{array}$$

And some others make no sense at all. Again, discussion about if this partial randomness in the system can be considered artistic or surrealist is not the purpose of this paper.

Other option is to try to apply *meta-rules*. We call *meta-rules* those rules which apply to constraints and modify them. Currently, there is no standard way of creating them, nor a global description of how they are defined. They have been designed by hand, and the authors are aware they are not necessarily the best ones. However, it can be interesting to show a high level example of them.

Let us suppose we have the domain which outputs the story in Section 3.2. In a particular state i,

$$s_i = \left\{ \begin{array}{cc} getin(criminal, bank) & catch(criminal, money) \\ go(criminal, warehouse) & chase(policeman, criminal) \end{array} \right\}$$

and C_i is still the same set than C_0 (the initial constraints set). Then, the exploration operators create a new fact: $friend(policeman, criminal)$. As we have the restriction $friend(?x, ?y) \wedge \neg chase(?x, ?y)$, the new fact would be considered as not coherent. However, we apply the next meta-rule:

> If there is some restriction about a fact, and the schema forbidding the inclusion of that fact is related with no more than one fact in the partial story, that schema can be removed.

Then, the $\lambda'(s_i, C_i)$ function returns:

$$\left\langle s_j = \left\{ \begin{array}{cc} getin(criminal, bank) & catch(criminal, money) \\ go(criminal, warehouse) & chase(policeman, criminal) \\ friend(policeman, criminal) & \end{array} \right\}, \right\rangle$$
$$C_j = C_i - \{friend(?x, ?y) \wedge \neg chase(?x, ?y)\}$$

Then, $shoot(policeman, criminal)$ is going to be added, but it is not coherent because there is a constraint against this inclusion:$shoot(?x, ?y) \wedge chase(?x, ?y) \wedge \neg friend(?x, ?y)$. The system can not find a meta-rule for modifying that constraint, so the new fact can not be added.

Finally, following the process, this is the story that has been obtained. It is coherent, and it was not possible to obtain it just by exploration:

$$\begin{array}{cc} getin(criminal, bank) & friend(policeman, criminal) \\ catch(criminal, money) & shoot(criminal, policeman) \\ go(criminal, warehouse) & kill(criminal, policeman) \\ chase(policeman, criminal) & go(criminal, warehouse) \end{array}$$

Meta-rules applied in CAST are useful for the examples and domains we are working on, but they can lead to incorrect stories (not coherent ones), if the definition of the meta-rules does not cover all necessary cases. These "necessary cases" are dependent on the domain, obviously: Sets of meta-rules covering adaptations for a given domain are useful only inside that domain. As it has been shown in the example, the rules are absolutely *ad hoc*, and further research is necessary to define them in a better way. This toy domain has been designed for exemplifying how the system works. It is possible, of course, to create more complex ones. There are many more possible definitions for λ'. The only purpose of explaining these results is to show that transformation is possible, and there are many possible ways for doing it. A generative system following these ideas can not forget the importance of the exploration (which have been done randomly in this article), because the results from exploration are the basis for transformation in a system like CAST. Next section discusses the main benefits, drawbacks and characteristics of this approach.

4 Discussion

CAST is intended to focus on transformation of constraint rules, and some possible approaches have been shown. The main question to ask is if CAST is a transformational creativity application. By trying to apply ideas present by Ritchie in [4] we can find three sets of generated objects (stories in this paper) about what is generated: The *conceptual space*, which would be in this case all "typical items", or stories following very obvious schemas; the *possible items*, or the full set of all possible stories; and all items in the *medium*, which contains all possible elements written with characters (if we consider the medium to be written text on paper, or on the computer screen).

Following Sharples [9] we have the *universe* (every possible object generated), and a set of *schemas*. They both create the *conceptual space*. Although both definitions for the *conceptual space* is not exactly equivalent, we can, for this discussion, relax the exact meanings, considering it to be the *expected stories*. It is difficult to define what "expected" means in this context, and for this discussion we assume an intuitive definition for it: stories which follow a very classical or typical plot, or character behavior.

We have seen, in Section 3.3, two examples of stories which could not be generated by the exploratory process in Section 3.2. However, it can be argued that this is a problem of the exploration algorithm, which is extremely simple. Of course it is, but it could be replaced with a more complex one and, in principle, the transformation processes could be applied. The point, then, is whether transformation in general, and the presented approaches as instances for it, are

useful, and really creative. Following the descriptions of what the *conceptual space* can be, it is changing. Bypassing the rules or modifying schemas really modify what can be generated, so the set of possible stories for the system is different. From this point of view, the system can be considered creative.

Relative to other systems, MINSTREL [7] also performs adaptation from knowledge present in a database. However, in MINSTREL, a TRAM is selected, and applied to a given narrative problem. Then, the narrative problem is mapped into the TRAM, and, applying it as a solution, it is "mapped-back" into the problem domain. CAST's transformations do not work in this way: Instead, CAST is able to apply any modification, not necessarily one related with a "similar" schema. Obviously, this approach can obtain a bigger set of new information, but can also produce bad results because it does not have any restriction beyond testing the validity, and the test functions are still not fully developed.

It is important to compare Sharples account for writing with CAST. In MEXICA Sharples' model directs the main algorithm. The main idea is that creative writing is produced by alternation of two processes: *engagement* and *reflection*, as explained in Section 2. CAST is not fully directed by Sharples' ideas, but it performs a sort of engagement and reflection process. The exploration generates knowledge without limits, and the transformation step checks for validity and adapts the constraints, accepts the new information or discards the content. Then, the exploration stage can be considered to be engagement, and it can be argued that checking the coherence and adapting the constraints set is a kind of reflection. However, this is not totally true, because in the checking stage, the information can be totally discarded, which does not literally follow Sharples' model.

Are meta-rules more useful than defining better constraint rules? It is more difficult, in general, to define meta-rules than defining constraint rules. On the other hand, constraints rules are domain dependent and meta-rules are not (if they are well defined, of course). The size of the set of valid stories that can be generated (which can be considered a measurement for the quality of the system) depends both on the constraint rules and on the meta-rules. It can be concluded, then, that meta-rules, covering general knowledge about how to transform, are useful for the system as itself, only if it is possible to create valid ones (general for many domains, applicable for every one of them). If this is true, the constraints database can be really domain dependent, and would not have to store general content. It is necessary further research and discussion to claim a formal, well demonstrated conclusion.

This paper's proposal is about trying a new option for this objective: Transforming story generation rules. It does not substitute the exploratory methods, (in fact, it depends on them), it just allow for a new, additive option. Although it is difficult to create meta-rules for this to allow a storytelling system create *exceptional* stories, it gives a new option. Even if, for a single generated story, it creates a new single, useful schema, the transformation can be worth the extra processing.

5 Conclusions and Future Work

CAST, a new storytelling system based on exploration and transformation has been presented and some results of the system are shown on the paper. The results of an example story is also explained in the paper, and the relation with transformational creativity is discussed.

Although this paper is focused on transformation of the rules, exploration can not be left as an unimportant matter. Choosing appropriate content for adding to the story can make the transformation more powerful, because probably it would not have to take into account completely meaningless additions. Next versions of CAST will be focused on better exploration approaches and, of course, a more deep study about transformation options.

References

1. Boden, M.: Computational models of creativity. Handbook of Creativity (1999) 351–373
2. Boden, M.: Creative Mind: Myths and Mechanisms. Routledge, New York, NY, 10001 (2003)
3. Bundy, A.: What is the Difference between Real Creativity and Mere Novelty? Behavioural and Brain Sciences (1999)
4. Ritchie, G.: The Transformational Creativity Hypothesis. **24**(3) (2006) 241–266
5. Wiggins, G.: Searching for Computational Creativity. New Generation Computing, Computational Paradigms and Computational Intelligence. Special Issue: Computational Creativity **24**(3) (2006) 209–222
6. Meehan, J.: The Metanovel: Writing Stories by Computer. PhD thesis (1976)
7. Turner, S.R.: Minstrel: A computer model of creativity and storytelling. Technical Report UCLA-AI-92-04, Computer Science Department, University of California (1992)
8. Pérez y Pérez, R.: MEXICA: A Computer Model of Creativity in Writing. PhD thesis, The University of Sussex (1999)
9. Sharples, M.: An account of writing as creative design. The Science of Writing (1996)
10. Gervás, P., Díaz-Agudo, B., Peinado, F., Hervás, R.: Story Plot Generation Based on CBR. Knowledge-Based Systems. Special Issue: AI-2004 **18** (2005) 235–242
11. Riedl, M.: Narrative Planning: Balancing Plot and Character. PhD thesis, Department of Computer Science, North Carolina State University (2004)
12. Bringsjord, S., Ferrucci, D.: Artificial Intelligence and Literary Creativity: Inside the mind of Brutus, a StoryTelling Machine. Lawrence Erlbaum Associates, Hillsdale, NJ (1999)
13. Lebowitz, M.: Creating characters in a story-telling universe. Poetics **13** (1984) 171–194
14. Gervás, P.: An Expert System for the Composition of Formal Spanish Poetry. Journal of Knowledge-Based Systems **14**(3-4) (2001) 181–188

Integrating a Plot Generator and an Automatic Narrator to Create and Tell Stories

Nick Montfort[1] and Rafael Pérez y Pérez[2]

[1] Program in Writing and Humanistic Studies
Massachusetts Institute of Technology
77 Massachusetts Avenue, Cambridge, MA 02139 USA
nickm@nickm,com
[2] Departamento de Tecnologías de la Información
Universidad Autónoma Metropolitana (Cuajimalpa)
Av. Constituyentes 1054, México D. F. 11950, México
rpyp@rafaelperezyperez.com

Abstract. We describe the integration of MEXICA, a well-developed story generation system which produces high-level plots, with the nn Narrator, an existing system for automatically narrating based on a simulated world and a plan with narrative specifications. In our initial integration of these systems in a pipeline, we did not seek to model the creative process explicitly. The initial integration nevertheless exposed issues of story representation and of the distinction between content and expression that are relevant to creativity. We describe these and discuss plans for a blackboard system that connects plot generation and narration more deeply.

Key words: Story generation, plot generation, narrative variation, narrating, system integration, story representation, action, event representation.

1 MEXICA-nn: Plot Generation plus Narrative Variation

Computer systems to generate stories are a fairly recent innovation, with the most significant early example (James Meehan's Tale-Spin) dating from the mid-1970s. Given this short history, such systems have been used for quite some time to understand creativity. Scott Turner's Minstrel, developed in the mid-to-late 1980s, was the first system to explicitly model creativity; it would move on to a new topic after a certain number of repetitions [1]. One system discussed in this paper, Rafael Pérez y Pérez's MEXICA, goes further and uses an explicit model of the creative process to reason about which plot elements will be included.

"Story generators," as they are often called, have been strongly focused on the content level (what happens in the story) rather than on the level of expression (how the story is told). For this reason, they can be more precisely characterized as *plot generators*. Some work has been done on shaping the expression of a story: Pauline

by Eduard Hovy [2] is an important early example. The other system discussed in this paper, Nick Montfort's nn, focuses more closely on variations in discourse that are specific to narrating. Some systems can work to some extent at both the story and discourse level, but we know of no previous efforts to integrate a well-developed story generation system and a well-developed system for narrating. This seems to be a significant oversight, given that literary study and narrative theory have elaborated for decades on how important both the story level and discourse level are for interesting storytelling.

In this paper, we describe the initial integration of MEXICA and the nn Narrator, systems that are each specialized to deal with one of the two essential levels of story. The two systems have been incorporated as subsystems of MEXICA-nn in a pipelined architecture. A plot representation originates from MEXICA, is handed to a middleware component that adds some information and transforms it into the type of representation required by nn, and, with the use of a manually specified plan for narrating, the story is finally told in a specific way by the nn Narrator.

We intend this as a first step toward effective and aesthetically pleasing automatic story generation which employs automatic narration. However, neither of us believes that the current pipelined scheme is really a good representation of the creative process. We believe it is very rare for fiction writers and other creative storytellers to develop everything at the story level first, in complete isolation from the language of the discourse level, and to determine how the telling will be done only at a later step — never revising, never reimagining. Our ultimate goal is to integrate these two systems in a different way, using a blackboard architecture. This would allow the plot representation and plan for narrating to be updated by either MEXICA or nn. Although MEXICA would primarily work at the story or plot level and nn at the discourse or expression level, both subsystems could suggest changes and collaboratively develop the final narrative. While we have learned some interesting things in developing our initial pipelined system, we hope that it will mainly be useful because it will inform this future work.

2 The Plot Generator MEXICA

The subsystem that deals with plot generation is MEXICA [3]. It was inspired by the engagement-reflection (E-R) cognitive account of writing as creative design [4]. MEXICA produces plots of stories about the Mexicas, the inhabitants, in centuries past, of what is now México City. (These people are sometimes inaccurately called the Aztecs.) During engagement the system generates sequences of actions guided by rhetorical and content constraints; during reflection, the system breaks impasses, evaluates, and, if necessary, modifies the material generated so far. Then, the system switches back to engagement and the cycle continues until the narrative is finished.

Creativity is one of the most astonishing characteristics of humans beings. It is a research area in the intersection of disciplines including cognitive science, artificial intelligence, psychology, and design. The core motivation behind the engagement-reflection model of computational creativity that informs MEXICA is to contribute to the study of the creative process, in particular in the area of plot generation.

One of the main problems in plot generation is the production of coherent sequences of actions. Traditionally, storytellers have dealt with this by employing explicit goals of characters and predefined story structures. These approaches helped to reduce the coherence problem. But they required that core parts of the stories be defined by the designers beforehand rather than being generated by the system. As a result, these programs and their outputs suffered from rigidity. MEXICA is an answer to the necessity of more flexible plot generators.

MEXICA is formed by two main blocks: the construction of knowledge structures (the K-S module) and the generation of plots through engagement-reflection cycles (the E-R module). The K-S module takes as input two text files defined by the user: a dictionary of valid story-actions and a set of stories known as the Previous Stories. The dictionary of story-actions includes the names of all actions that can be performed by a character within a narrative along with a list of preconditions and postconditions for each. In MEXICA all preconditions and postconditions involve emotional links between characters (e.g., jaguar knight hates the enemy). The Previous Stories are sequences of story actions that represent well-formed narratives. With this information the system builds its knowledge base. The E-R module takes two main inputs: the knowledge-base and an initial story-action (e. g., the princess heals jaguar knight) that sets in motion the E-R cycle. Once the E-R cycle ends the system perform the final analysis, a process that touches up the generated plot.

MEXICA is implemented in Pascal/Delphi. The system includes a text editor where the user defines in text files the dictionary of valid story-actions and the group of previous stories. To produce a new plot, the system provides an interface where the user introduces an initial action that triggers the engagement-reflection cycle. MEXICA creates two reports: 1) a detailed trace of the engagement-reflection cycle, which includes the final plot; 2) a representation of knowledge structures. Both reports can be visualized within the program. The implementation includes 21 modifiable parameters that control different aspects of the E-R cycle, allowing the user to experiment with the model.

MEXICA shows that the E-R model of computational creativity allows the generation of novel and interesting plots [5]. The program uses a representation of emotions to associate possible actions and continue the story in progress. In this way, MEXICA contributes to the study of the role of emotions during the creative process [6]. The program includes a process to evaluate the interestingness of the story it creates. As far as we know, no other plot generator is capable of this type of evaluation.

3 The nn System and Its Narrator Module

The subsystem that deals with the expression level is nn, a system originally created to allow the implementation of interactive fiction (that is, text adventure games) with narrative variation [7]. The components of nn that are specific to interactive fiction are not used in MEXICA-nn; the module that is used is one that has been central to research in narrative variation: the Narrator.

Enabling new types of creative writing and programming is the main purpose of nn. Specifically, nn is designed as an interactive fiction system, to enable general, simple manipulation of the discourse level. It is already possible to place characters and objects in various locations in an interactive fiction worlds, to animate them and have them react. The eventual release of nn will hopefully provide author/ programmers of interactive fiction with discourse-level facilities that are similar to the existing story-level capabilities of state-of-the-art systems such as TADS 3 and Inform 7. While much work in IF has dealt with creating intelligent or emotional characters, nn seeks to do something different: to create more interesting narrators.

For the purposes of integrating MEXICA and nn, many modules of nn have been left out. The ones dealing with tokenizing and parsing input, for instance, have been omitted, as there is no user input in MEXICA-nn. The one that simulates the fictional world is also not necessary, because in MEXICA-nn the events of the story are simply provided by MEXICA and do not need to be simulated within an interactive fiction framework. Others dealing with specialized functions that do not apply to the narrating task (such as the "meta" operations of *restart*, *undo*, *quit*, and so on) have also been omitted. What is left is the core module of the nn research project, the Narrator, and the two representations that are essential to narrating: the focalizer worlds and the plan for narrating.

The Narrator module has an internal architecture. It is structured as a three-stage pipeline, using a model that is typical of natural language generation systems. In the reply planner, the high-level structure of the reply (or, in the case of MEXICA-nn, of the entire output document) is determined. The narrative parameters and the reply structure determined in this first stage are sent to the microplanner, which determines the grammatical specifics based on those inputs. Finally, the detailed specification for output is converted into text by the realizer, which does lexicalization.

nn is implemented in Python 2.5. To create a work of interactive fiction, an author/ programer writes a game file which is simply Python code that makes use of pre-defined classes and assigns values to certain expected variables. The existents in the fictional world are represented as a dictionary of objects (each of class Actor, Room, or Thing) and the actions as a list of Action objects, each of which contains one or more lower-level Event objects.

Several important types of narrative variation have been modeled in the system. Specifically, one significant aspect of narrative variation has been implemented within each of the five major categories described in *Narrative Discourse* [8]. New

theoretical ideas have been developed as a result of these implementations. One is that in generating text, order and time of narrating must be considered jointly even though Genette has placed them in separate categories. Another is that simple sequences are inadequate to represent order, while an ordered tree provides a good representation. Other results pertain to narrative distance (which may be best seen as a composition of other effects) and the worlds representing perception by focalizers (which are fundamentally different from the simulated world).

4 A First Pass at Connecting the Systems

A MEXICA-nn narrative is generated by running a modified MEXICA to generate XML as defined by the MEXICA plot schema. An invocation of the middleware component of the system then takes this plot representation and elaborates it, transforming it to a different XML representation of existents and actions that conforms to the nn schema. Finally, nn is invoked with a specific plan for narrating to produce a narrative.

The middleware maps high-level plot elements, as output by MEXICA, to more specific sets of actions, each of which must include a sequence of one or more events. For instance, the MEXICA action `FAKED_STAB_INSTEAD_HURT_HIMSELF` is mapped to two actions in nn; the first one has FEINT as its verb and the second one has INJURE as its verb. The FEINT action has a MISC event within it. This is the type of event used to represent something that does not have a physical consequence or involve the application of force. The INJURE action is best represented with a IMPEL event (representing the character's application of force to himself) followed by a MODIFY event (representing that the character's health was affected). These detailed events can come into play if the speed of narration is slowed down and every detail of what happens is being represented by the narrator.

In one run, the modified version of MEXICA used as a subsystem of MEXICA-nn produced the following XML representation of a plot:

```
<story>
 <existents>
  <location name="Lake"/>
  <location name="City"/>
  <character name="HUNTER" gender="male"  location="Lake" />
  <character name="JAGUAR_KNIGHT" gender="male"  location="Lake" />
  <character name="VIRGIN" gender="female"  location="Lake" />
 </existents>
 <actions>
  <action verb="ACTOR" agent="JAGUAR_KNIGHT" time="1" />
  <action verb="ACTOR" agent="VIRGIN" time="2" />
  <action verb="ACTOR" agent="HUNTER" time="3" />
  <action verb="WERE_RIVALS" agent="JAGUAR_KNIGHT" object="HUNTER"
   time="4" />
  <action verb="FAKED_STAB_INSTEAD_HURT_HIMSELF"
   agent="JAGUAR_KNIGHT" object="HUNTER" time="5" />
  <action verb="CURED" agent="VIRGIN" object="JAGUAR_KNIGHT"
```

```
   time="6" />
  <action verb="MUGGED" agent="JAGUAR_KNIGHT" object="VIRGIN"
   time="7" />
  <action verb="WENT_TENOCHTITLAN_CITY" agent="JAGUAR_KNIGHT"
   NewLocation="City" time="8" />
  <action verb="HAD_AN_ACCIDENT" agent="JAGUAR_KNIGHT" time="9" />
  <action verb="DIED_BY_INJURIES" agent="JAGUAR_KNIGHT" time="10" />
 </actions>
</story>
```

This representation was translated by the middleware into an elaborated XML representation that used the nn schema and specified finer-grained events. The modified version of the nn Narrator that was used was then able to produce a variety of different narrations. This one was accomplished using chronological order, setting the temporal position of the narrator to a point before the events (hence, the narration takes place in the future tense), and adding a filter that expresses surprise:

The jaguar knight will oppose the hunter!
Meanwhile, the hunter will oppose the jaguar knight!
Then, the jaguar knight will feint at the hunter, dude!
Then, the jaguar knight will strike himself!
Then, the virgin will heal the jaguar knight! Awesome!
Then, the jaguar knight will threaten the virgin, man! Awesome!
Then, the jaguar knight will rob the virgin!
Whoa, then, the jaguar knight will go to city!
Then, the jaguar knight will slip!
Then, the jaguar knight will fall!
Then, the jaguar knight will die!

The next narration was produced using settings for retrograde order (so that the most recent event is narrated first) and with time of narrating set to be simultaneous with the last event. Although this is a simple rearrangement out of chronological order, it shows the generation of different tenses that are necessary for more nuanced orderings, such as occasional flashbacks. The final action, told first, is narrated in the present tense and the ones leading up to it are told in the past tense:

The jaguar knight dies.
Before that, the jaguar knight fell.
Before that, the jaguar knight slipped.
Before that, the jaguar knight went to city.
Before that, the jaguar knight robbed the virgin.
Before that, the jaguar knight threatened the virgin.
Before that, the virgin healed the jaguar knight.
Before that, the jaguar knight struck himself.
Before that, the jaguar knight feinted at the hunter.
Before that, the hunter opposed the jaguar knight.
Meanwhile, the jaguar knight opposed the hunter.

A straightforward present-tense narration can be generated where the narrator speaks hesitantly (a stylistic variation that isn't actually specific to narrative) and where all the details of each action are told as slowly as possible:

> The jaguar knight, um, starts, er, to rival the hunter. The jaguar knight, um, opposes the hunter.
>
> Meanwhile, the hunter starts to rival the jaguar knight. The hunter opposes the jaguar knight.
>
> Then, the jaguar knight starts to feint the hunter. The jaguar knight, um, feints at the hunter.
>
> Then, the jaguar knight starts to injure himself. The jaguar knight strikes himself.
>
> Then, the virgin starts to, er, cure the jaguar knight. The virgin heals the jaguar knight.
>
> Then, uh, the jaguar knight, um, starts to threaten, uh, the virgin. Uh, the jaguar knight, er, threatens the virgin.
>
> Then, the jaguar knight starts to rob the virgin. Um, the jaguar knight robs the virgin.
>
> Then, the jaguar knight starts to go. The jaguar knight goes to, um, city.
>
> Then, the jaguar knight starts to, uh, slip. The jaguar knight slips.
>
> Then, the jaguar knight starts to fall. The jaguar knight falls.
>
> Then, the jaguar knight starts to, uh, die. The jaguar knight dies.

5 Differences in Representation of Action and Story

Some important differences between MEXICA's representation and the nn Narrator's were revealed by integrating the two systems. nn combines a representation of actions and existents with a plan for narrating, while MEXICA, focused on the plot level, lacks the latter type of representation. But nn's representation of actions and events is also quite different from MEXICA's. One mismatch is that no Things (the class used in nn for inert items or props) are represented explicitly by MEXICA; it is simply implicit in an action involving stabbing that the character undertaking the action has an edged weapon of some sort by the time an action is carried out. The character may possess that weapon to begin with or acquire it in the course of the story. Which of these happens and how exactly it happens needs to be added during elaboration for nn's Narrator to able to narrate in the best possible way.

Additionally, the granularity of action is different in the two systems. A single MEXICA plot element may correspond to multiple high-level actions in nn. For instance, a reflexive MEXICA action involving A and B, such as `FIGHT_EACH_OTHER`, corresponds to two simultaneous actions in nn, one with A as agent and one with B as agent. A plot element that corresponds to a compound action maps to a sequence of actions that occur one after the other in nn. Each of the low-

level events within these actions must also be specified to indicate each blow of the battle. nn's more numerous and finer-grained actions are difficult to specify, but when they are in place, they are very useful for creating narrative variation of different sorts, since each event may be included or excluded and, if told at all, may be told more or less rapidly. Some of the work of mapping MEXICA's plot representation to an nn story representation is straightforward; other work involves creativity and authorship. The more creative work involves determining what specific sequence of blows occurs in a fight, what weapon a character uses to attempt to stab another character, and how a particular weapon or other object is acquired by a character. In the future, we plan to develop MEXICA-nn to accomplish these things not with fixed mappings but rather in a way that is consistent with MEXICA's model of the creative process.

On the other hand, the nn Narrator's representation is not more detailed in every way. MEXICA uses a model of character's emotional affinities for one another in determining the plot. This model could be used in narrating, too. To provide a simple example: a narrative could be produced in which actions are elided or passed over very quickly when they are undertaken by characters that jaguar knight dislikes. The resulting narrative would favor jaguar knight by emphasizing the actions of his friends, or, more precisely, the people who he likes at that point in time. MEXICA also uses particular techniques to generate interesting plots. Unlikely plot elements are proposed by the system and, using abductive reasoning, the system tries to find explanations for them in the form of other plot elements. For instance, if the princess loves jaguar knight at one point and then later intentionally kills him, it may be because she discovered that he killed her father. If information about the plot generation process was available to the Narrator, the narration of the story could follow the trace of the creative process, describing the unlikely event first and then, in a flashback, explaining why it came to pass.

As we have discussed, the type of representation used is substantially different for the nn Narrator and for MEXICA. To some extent, one system might benefit from having a finer-grained representation that is motivated by the other. But this is unlikely to be the best answer in every case. In many ways, MEXICA takes a higher-level view of actions and existents for a good reason. A complete computer model of story generation, from plot generation to narrating, is likely to require different types of knowledge structures at different levels of abstraction (or, knowledge representations that can be easily manipulated at different levels of abstraction) interacting together in order to generate its final output.

MEXICA employs three types of knowledge representations at three different levels of abstraction: the Tensional representation (graphics of the dramatic tension of the story in progress), the Atoms (groups of emotional links and tensions between characters) and the Concrete representation (a copy of the story actions and the previous stories). The Concrete representation includes more precise information than the Atoms; and the Atoms include more precise information than the Tensional

representation. The engagement-reflection cycle employs these three knowledge structures during plot generation.

The nn Narrator, on the other hand, deals with a representation of the telling that is absent from MEXICA (the plan for narrating), a representation of characters' perceptions (the focalizer worlds), and a representation of the "real world" as far as the fiction is concerned (the world model, with actions and existents). The last of these is the point of overlap with MEXICA, but even then, as we have described, the way actions are represented is not the same and the same existents are not present in both systems.

We find it appealing to develop a system that goes upwards and downwards through concrete and abstract knowledge representations during the weaving and telling of a story. A more general representation of an action (e.g., jaguar knight and the enemy fight each other) is suitable during initial enplotment, when a general framework for the narrative is being put together; a detailed description of a fight (e.g., types of weapons used, particular blows) allows the system to very the telling to a greater extent, since there are more choices of what to omit and what to include and the actions can be arranged in a wider variety of ways. As we continue development, we will look to develop a system that employs representation at both levels to generate a plot and a particular telling. The middleware described in this paper is the first step in this phase of the work.

6 Implications and Future Work

There are other steps to be taken beyond modification of knowledge representations and a better pipelined integration. Currently, when MEXICA-nn is invoked the user specifies a particular plan for narrating and the pre-defined parameters of that plan are simply applied to the story. A better approximation of the creative process would involve having MEXICA-nn choose a plan that is appropriate to the story material. For instance, a story of a character's steady downfall might be narrated chronologically, while a story with many reversals might be told using flashbacks to connect the most interesting actions with each other.

We hope that plans for narrating can eventually be developed at a deeper level, co-evolving with the plot. In a blackboard system, nn could propose a plan for narrating of a specific sort — for instance, sylleptic narration in which everything that happens in the city is told first, then everything that happens in the forest, and then everything that happens by the lake. nn could request that a larger number of interesting actions occur in the forest. MEXICA could then work to satisfy this request, either by changing some plot elements so that they occur in different location or by adding plot elements. Similarly, MEXICA could register requests regarding the plan for narration, asking that plot elements be told in a certain order or that certain existents or plot elements be emphasized.

This paper has described the initial integration of MEXICA and the nn Narrator to produce a complete computer based storyteller, MEXICA-nn. This work has led us to reflect on the importance of manipulating represented knowledge at different levels of abstraction when developing computer models of creativity. We hope this paper encourages people to integrate systems. In the short term, it is a source of important information about meaningful differences between systems that might be difficult to obtain in another way. In the long term, we believe integrating well-developed systems will be essential in tackling complex problems such as the invention and telling of stories, which require specialized creative processes that must also interact and depend upon one another.

References

[1] Wardrip-Fruin, N. (2009). *Expressive Processing*. Cambridge: MIT Press.

[2] Hovy, E. (1988) *Generating Natural Language under Pragmatic Constraints*. Hillsdale, N.J.: Lawrence Erlbaum, 1988.

[3] Pérez y Pérez, R. & Sharples, M. (2001) MEXICA: a computer model of a cognitive account of creative writing. Journal of Experimental and Theoretical Artificial Intelligence. Volume 13, number 2, pp. 119-139

[4] Sharples, M. (1999). *How we write: Writing as creative design*. London: Routledge.

[5] Pérez y Pérez, R. & Sharples, M. (2004) Three Computer-Based Models of Storytelling: BRUTUS, MINSTREL and MEXICA. Knowledge Based Systems Journal. Vol. 17, number 1, pp. 15-29.

[6] Pérez y Pérez, R. (2007). Employing Emotions to Drive Plot Generation in a Computer-Based Storyteller. Cognitive Systems Research. Vol. 8, number 2, pp. 89-109.

[7] Montfort, N. (2007). "Generating Narrative Variation in Interactive Fiction." PhD dissertation, University of Pennsylvania. <http://nickm.com/if/Generating_Narrative_Variation_in_Interactive_Fiction.pdf>

[8] Genette, G. (1980). *Narrative Discourse: An Essay in Method*. Trans. J. E. Lewin. Ithaca, NY: Cornell.

A Framework for Building Creative Objects From Heterogeneous Generation Systems *

Carlos León[1], Jorge Carrillo de Albornoz[2], and Pablo Gervás[3]

Departamento de Ingeniería del Software e Inteligencia Artificial
Universidad Complutense de Madrid
[1]cleon@fdi.ucm.es, [2]jcalbornoz@fdi.ucm.es, [3]pgervas@sip.ucm.es

Abstract. There exist several types of generative systems which produce some artistic output not necessarily considered to be creative. However, the processes they follow can, in principle, be combined between them, possibly obtaining a creative product. Although nowadays giving a general description of how such systems must be implemented is far from being possible, this paper presents a framework for putting together generative systems in such a way that the output can be creative. The paper shows preliminar ideas and discusses the main conclussions obtained from them.

Keywords: Computational Creativity, Cinematographic Direction, Storytelling, Framework, Artificial Intelligence.

1 Introduction

Building generative systems which are not creative for artistic production is not difficult. For instance, rule based systems controlling a story generation program can be easy implemented, but, probably, the output will not be as creative as needed for that program to be considered really useful in general. The same ideas apply for any other generative system (like automatic camera direction).

However, complex creative objects, like movies, are composed of many heterogeneous systems, and for these systems to be considered creative, it is not necessary that every part of the creation follows a creative process.

Given these characteristics, to study how different processes are connected in a full complex system becomes interesting and useful in the field of the Computational Creativity, where computers have to follow models of creativity in order to obtain new objects that *would be deemed as creative if found in humans* [1]. This knowledge would allow us to put several heterogeneous systems together, bringing new possibilities in the productions of such systems.

* This research is funded by the Ministerio de Educación y Ciencia (TIN2006-14433-C02-01), Universidad Complutense de Madrid, Dirección General de Universidades e Investigación de la Comunidad de Madrid (CCG07-UCM/TIC 2803) and a grant from Caja de Burgos, Jóvenes Excelentes 2008 contest.

However, obviously it is extremely difficult to give a general description of how to put generative systems together. In fact, the authors are not sure if such description exist: probably it is not possible to define a general way of combining them. Basically, what we can do by now is to look for *ad-hoc* solutions, trying to figure out what functions are to be solved, and trying to give a valid solution to them. Or, at least, a valid solution for the domain or the particular application that is being developed.

In this paper a case-study of a framework for one of these systems is presented, showing how an application that creates 3D short stories, using four different systems could be created: an automatic story teller, and module that adapts stories to given cinematographic schemas, an automatic camera positioning system, and a camera motion director, as depicted in Figure 1. The focus of this particular research has not been put on the systems separately, but on the nexus between the four, to study how to put several heterogeneous systems together in order to obtain a creative scene from these not necessarily creative systems.

This study is oriented to the definition of a system for creating 3D movies called Spiel, which will try to implement all these ideas in order to get a working creative system.

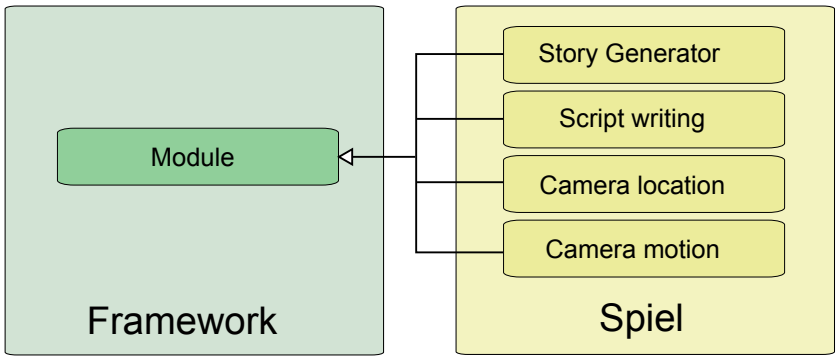


Fig. 1. Spiel's modules.

2 Previous Work

Computational Creativity tries to study processes related to human creative production, and is heavily influenced by Boden's work [2, 3]. The main issue about how to build creative systems is that *creativity* is not precisely defined [4]. However, building systems that could be considered creative is still possible and interesting.

There are several approaches to heterogeneous systems for cinematographic creation, but most of them are not focused on Computational Creativity. We can cite, however, some creative automatic systems from which several ideas have been taken to carry out the development of Spiel.

Christianson et al. [5] use film idioms to adapt cinematographic techniques for portraying 3D animations. They formalise film idioms with their *Declarative Camera Control Language* (DCCL). As input the camera system use an animation trace representing the action that has already occurred and a number of sequences indicating which actor to film over each time. The system presented in [6], *The Virtual Cinematographer*, is a camera control system for a real-time virtual party that also use the concept of film idioms. The input is obtained directly from the state of the environment to choose the best idiom according to the actions. Each idiom is implemented as a finite state machine that describe the cinematographic techniques, which cover the current situation.

In the same line, the work presented in [7] shows a real-time system for interactive narration in virtual worlds. In [8] a similar approach is explained in a real-time computer game system. All this research is not focused on the creative processes involving the creation and visualization of a story, or in how the nexus between these components can enrich the interaction with humans.

Charles et al. [9] propose a system that combines an interactive storytelling system with an automatic camera system based on cinematographic techniques, which decides the correct idiom covering the actions of the characters, giving different weights according to the relevance in the situation.

Storytelling follows, however, a different research tradition. Creativity has been studied in several storytelling systems (MEXICA [10] or Minstrel [11]). Results from those programs have been very influential on the development of Computational Creativity. They try, in general, to follow psychological models of human creativity, trying to apply a computational model of them to the generation of stories. But there also are other systems, (like TaleSpin [12], Fabulist [13] or Universe [14]), which, even creating good stories, are not focused on creative processes.

Next sections explain how this can be done: even without using creative systems, it is possible to build a creative object, by the interaction of the generative programs.

3 A Framework for Building Artistic Objects from Heterogeneous Modules

With all these ideas, a framework which tries to study how to automatically create a complex artistic object, like a 3D movie, from not necessarily creative systems could be possible, and an instantiation of this framework (Spiel), are presented. This section explains in detail how the global system carries out the task of making several different heterogeneous systems work together. First an overview of the system, explaining the algorithm, is given, then some important details of the modules which instatiate the framework are explained.

3.1 How the Framework Works

The framework for creating artistic objects (mainly focused on movies) from heterogeneous modules is not a story generator, nor a camera direction system: it is a software whose application is focused on using several systems together, to create new information which the modules themselves could not create.

It is really difficult to design a system capable of carrying out this task for every possible set of systems. It would be necessary to define a language which every module should speak in order to intercommunicate those modules between them and with the main system. Also, functions for deciding operations like "when to apply each module" or "which operation must be done" should be defined in such a way that they would cover all possible types of modules, by giving a general description of those functions, or by covering all cases.

Although this is, in principle, possible, we are far from being capable of creating such a system. Instead, the logical approaches should follow a kind of *ad-hoc* solutions, trying to gather all possible information in order to gaining knowledge and experience for, some day, being able to develop a general system.

SPIEL, the framework's instantiation, is one of such *ad-hoc* systems. However, it is not only intended to cover a specific case, but it is fully focused on creating a "growing" nucleus, able to use, in the near future, different types of modules (stories, voice, 3D environments, plain text...). Basically, the framework's algorithm follows the schema shown in Algorithm 1. It is a very simplistic approach to an opportunist architecture. Several modules are present in the system, and the object which is going to be generated (a movie in the SPIEL instantiation), is being created step by step, by the sequential operation of the modules.

For directing what must be generated, modules are initialized with data guiding the generation. For instance, a story generation module should be initialized with constraints about what is to be created (a love story, perhaps).

What this algorithm is doing is explained in the list below. The algoritm is executed with a list of modules which implement the functions `validity`, `appropriate`, `new-solution` and `apply-solution` (what these functions are is explained), and it is assumed that the system's status is the partially built movie:

- **Lines 1-4**. The object which is going to be generated is initialized (s_0), and, running the `validity` function, each module yields a pair of values: the value for $moduleEvaluation_i$ is a real number in the interval [0..1], which is a measure of what the module thinks of the partial creation of the system (the partially generated movie). 0 means the module "thinks" the partial object is completely wrong, and 1 means that the partial object is valid enough to be the solution. This value is computed by each module in a different way, and considering different aspects of the generated object. The $moduleProblem$ value is a description of what the module considers to be wrong. This description is dependent on the particular instantiation of the system. The different incomplete parts of the partial object are defined in these steps.
- **Line 5**. After the system has computed every $\langle moduleEvaluation, moduleProblem \rangle$ pair of each module by calling its `validity` function, the framework applies

Algorithm 1 Using several modules in the framework for creating a single object

1: $s_0 \leftarrow$ *initial object*
2: **for all** $module_i$ in *modulelist* **do**
3: $\langle moduleEvaluation_i, moduleProblem_i \rangle \leftarrow module_i.validity(s_0)$
4: **end for**
5: $\langle evaluation, problem \rangle \leftarrow$ select the *most important pair* from the set of pairs $\langle moduleEvaluation_n, moduleProblem_n \rangle$
6: **while** $evaluation \neq$ valid **do**
7: **for all** $module_i$ in *modulelist* **do**
8: $moduleAppropriate_n \leftarrow module_i.appropriate(problem)$
9: **end for**
10: *chosen* $\leftarrow$ module with $max(moduleAppropriate_n)$, or *none* if no module is appropriate
11: **if** it has been possible to find a *chosen* module **then**
12: apply *chosen.newsolution*() to s_j
13: **if** s_j has new information **then**
14: **for all** *othermodule* in *modulelist* **do**
15: $othermodule_i.applysolution(s_j)$
16: **end for**
17: **for all** $module_i$ in *modulelist* **do**
18: $\langle moduleEvaluation_i, moduleProblem_i \rangle \leftarrow module_i.validity(s_{j+1})$
19: **end for**
20: $\langle evaluation, problem \rangle \leftarrow$ select the *most important pair* from the set of pairs $\langle moduleEvaluation_n, moduleProblem_n \rangle$
21: **else**
22: discard the currently *chosen* module, and choose a different one in the next iteration
23: **end if**
24: **else**
25: exit program
26: **end if**
27: **end while**

a function, which can be defined in many ways, which returns the global system $\langle evaluation, problem \rangle$ pair. This function could return the problem from the module with the lowest *moduleEvaluation* value (which would be the pair with the "hardest" problem). Or it could return the mean value from the *moduleEvaluation* values of each module, and then a *problem* randomly chosen from the different modules. A discussion about some possible approaches is presented in Section 4. After this step, the problem (the incomplete part) which has to be solved is defined.

- **Line 6**. The system then checks whether the *evaluation* value is good enough to be considered valid. There are several possible ways of computing this boolean expression. Possibly, the most straightforward one is to check that the real value of *evaluation* is over a given threshold. This line guides a loop which will end once the object is created, or when it is considered impossible to create an object with the requisite from the user.
- **Lines 7-10**. Once we have the *problem* which has to be solved, each module runs its `appropriate` function. This function returns a real number in the interval [0..1], 0 meaning that it will not be able to solve a particular *problem* object, and 1 meaning it can perfectly solve the problem. Obviously, it is difficult to know whether a module can solve a problem before trying to solve it. What the `appropriate` function returns is a "possibility" value ($moduleAppropriate_n$). For instance, if the *problem* object returned by the computation of the `validity` function stores information about missing information of a story, a story generation module will return a higher value for the `appropriate` function than a module managing camera motion. However, even selecting a particular module, nothing ensures that it will be able of really solving the issues present in the *problem* object.

  After gathering all possible values for `appropriate` from each module, it is necessary to decide which module will be chosen. Initially it could be a good option to choose the module with higher `appropriate` value, but this value is only an approximation, and it can lead to local-minimum. That is why the framework adds random noise, and chooses, in some iterations (the frecuency of this noise can be parametrized), a module which is not the one with highest `appropriate` value. Section 4 discusses the benefits of this approach from the creativity point of view.

  If no module can be chosen (for instance, every module returns 0 as the value of the `appropriate` function), the system exits with a failure vaule: It has not been possible to create an object with the initial constraints.
- **Lines 11-12**. If it has been possible to find an appropriate (*chosen*), module, this chosen module applies its internal processes (what the module is really created for: a story, a new camera layout...), and generates a new object s_j applying its `new-solution` function (which returns partial information which completes the story: in SPIEL, a new set of facts for the story, for instance). If the `new-solution` function is not able to return a value, the algorithm will try to choose a different module. If no other module is *appropriate* (as explained in the previous list item), the generation fails.

- **Lines 13-16**. If the partial object (s_j) has been updated, The `apply-solution` function is called in each module. This function receives the description of the *solution* object just created by the choosen *module*, and updates the description of the partial whole object. For instance, if the chosen *module* was the story generator module, and it created three new facts for the story (which is only a part of the movie), the camera direction module would receive those three facts, and would update its information about the story. Obviously, this function assumes that every module knows how to apply information from other modules. So, although they can be heterogeneous, they can not be totally independent of each other.
- **Lines 17-20**. Finally the system checks again for the validity of the new partial object s_{j+1}.

The overall benefits of the approach followed in this algorithm are discussed in Section 4. Section 3.2 explains in detail how the previously commented functions are really instantiation.

3.2 Spiel: An Instantiation for Creating Movies

As it has been said before, four independent modules have been designed to instantiate this framework. They have been designed for interoperability. The next list details information about them, focusing on the instantiation of previously defined four functions for interacting with SPIEL, and, therefore, between them. These modules are not creative by any means, however, we want to discuss whether the output from these not necessarily creative modules can be creative or not.

- The *story generation module* outputs a list of first order logic-like predicates in this form[1]: `character(aragorn)`, `go(aragorn, forest)`. For instantiating the `validity` function, it accepts a story as valid if it is coherent with the user input (using hand-made logic rules). For example, if the user input is `kill(aragorn, sauron)`, and the resulting story-state is just `character(aragorn)`, `go(sauron, mordor)`, the story will be considered to be non-valid.
  The `new-solution` function returns, taking into account the current state of generation (the partial story, and the partial camera layout), a new story with logic facts.
  The `appropriate` function returns a valid value if the module can find, in its facts database, the set of facts necessary for completing the story, given the current problem. Finally, `apply-solution` updates module's state if the solution is a new story, and if the solution is a new camera location, it performs a mapping between the camera state and a set of "emotional facts" (`emotion(love, intense)`) in the story to create a new "emotional" state.

[1] We are working in The Lord Of The Rings domain.

- The script writing module adapts stories to typical story scripts. If, for example, a romantic scene is lacking the final kiss, this module adds that information to the set of facts (which has the same format as the previous module's output). The instantiation for the four functions follows approximately the same behaviour that for the story creating system (the rules vary, but the semantic is the same), but for the `appropriate` function: it computes this value by computing the distance between the script schema and the generated story. The less the distance (without being equal), the more appropriate the application of this module. That is, if there is only one different fact between the story and the schema, this module can solve the problem.
- The camera positioning module abstracts the cinematographic knowledge in order to determine which position and orientation, for a specific action or event in a story, are the most suitable, and it generates a new camera layout, creating a structure according to the story compose of scenes and sequences and the appropriate camera configuration for each sequence. It calculates the `validity` function output by computing a *specificity* value: whether the story shows a clear emotion within a predefined set, or match with one of the sequence predefined in the module, or the sequence of the scene don't lie in a cycle of camera configuration. It considers itself to be appropriate or not depending on how specific (as defined before) the partial story is: The higher the specificity, the higher the `appropriate` value. Finally, `apply-solution` updates module's state to generate a new structure with their appropriate camera configuration, if one solution is proposed by other modules. As a `new-solution` this module propuse new information like a movement for some character, or an emotional intensity for a sequence, or to introduce a new kind of predefined sequence in the story. With this information the new state of the modules will be update and checked again for correctness.
- The camera motion module works in the same way that the camera positioning module does. However, the output is composed of camera movements (track, pan, tilt) instead of positions.

4 Discussion

The framework and Spiel are not creative systems (the main algorithm is quite simple): the processes they run are rule based, without any creative idea. What the framework does is to handle, in a kind of opportunist and distributed way, different heterogeneous processes. What we want to discuss, basing on the main conclusions we have obtained during the development of the framework and its instantiation, is whether a creative object can be obtained by applying this kind of processes (trying to combine several generative systems), how can creativity emerge.

Although there are different computational systems focused on generating creative objects (like stories), and whole systems for creating 3D movies (or parts of them), there is not much focus on whole systems trying to generate

creative 3D movies. Of course, SPIEL is far from being able of such a creation, and this paper just shows the first steps towards this ambitious objective. The main point of this text is to discuss whether this is possible, and what processes could lead to creativity.

The main point we can argue is that this kind of frameworks can lead to results which would not be the output of a sequential system (that is, creating a story, and, after that, creating a camera layout). Modules provide feedback for other modules, which should, in principle, lead to more "enriched" results.

For this to be possible, it is important to create a good definition for choosing which module to apply and a the evaluation functions for the system. If these functions are just focused on choosing which module returns the highest values, probably the output of the system will not be much better. However, the study of how collaborative decission for this functions can lead to emergent creativiy is interesting.

For instance, if the result of the global *evaluation* value (as defined in Section 3.1) is computed taking into account every module's result (perhaps computing the mean value), the heterogeneous systems collaborate, and the global system gets enriched by this process, in the sense that a new evaluation function emerges, and this function is different, and not previously designed by the module creator (although obviously it is previously conceived by the framework's operation).

5 Conclusions and Future Work

A framework for creating compound creative systems, has been presented, and an instantiation for it, SPIEL. It is fully focused on how independent systems can work together to build a complex creative object. The main conclusion is whether, even having obtained new data from the use of this system, these new data can be considered creative, or just a product of feeding modules with additional data.

Next versions of the framework and SPIEL will focus on creating more modules, trying to build a real 3D creation system. It is planned to create modules not dependent on the framework (and adding a wrapper for it). Letting the system follow a state-space search, like a backtracking (trying many different possible options in each iteration), would improve the system output.

Also, the greedy approach is only intended to show the preliminary possibilities of such a system, but a blackboard architecture, or a backtracking state space search could be interesting for this purpose, letting the modules communicate in a more powerful way.

References

1. Wiggins, G.: Searching for Computational Creativity. New Generation Computing, Computational Paradigms and Computational Intelligence. Special Issue: Computational Creativity **24**(3) (2006) 209–222

2. Boden, M.: Computational models of creativity. Handbook of Creativity (1999) 351–373
3. Boden, M.: Creative Mind: Myths and Mechanisms. Routledge, New York, NY, 10001 (2003)
4. Ritchie, G.: The Transformational Creativity Hypothesis. New Generation Computing, Computational Paradigms and Computational Intelligence. Special Issue: Computational Creativity **24**(3) (2006) 241–266
5. Christianson, D.B., Anderson, S.E., wei He, L., Salesin, D., Weld, D.S., Cohen, M.F.: Declarative camera control for automatic cinematography. (1996) 148–155
6. wei He, L., Cohen, M.F., Salesin, D.H.: The virtual cinematographer: a paradigm for automatic real-time camera control and directing. In: SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, New York, NY, USA, ACM (1996) 217–224
7. Amerson, D., Kime, S.: Real-time Cinematic Camera Control for Interactive Narratives. In: Working Notes of the AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment, Stanford, CA, USA, AAAI Press (March 26-28 2001) 1–4
8. Lin, T.C., Shih, Z.C., Tsai, Y.T.: Cinematic Camera Control in 3D Computer Games. In: Proceedings of 12th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG 2004), University of West Bohemia, Campus Bory, Plzen-Bory, Czech Republic (February 2-6 2004) 289–296
9. Charles, F., Lugrin, J.L., Cavazza, M., Mead, S.J.: Real-time camera control for interactive storytelling. In Mehdi, Q.H., Gough, N.E., eds.: GAME-ON. (2002)
10. Pérez y Pérez, R.: MEXICA: A Computer Model of Creativity in Writing. PhD thesis, The University of Sussex (1999)
11. Turner, S.R.: Minstrel: A computer model of creativity and storytelling. Technical Report UCLA-AI-92-04, Computer Science Department, University of California (1992)
12. Meehan, J.: The Metanovel: Writing Stories by Computer. PhD thesis (1976)
13. Riedl, M.: Narrative Planning: Balancing Plot and Character. PhD thesis, Department of Computer Science, North Carolina State University (2004)
14. Lebowitz, M.: Creating characters in a story-telling universe. Poetics **13** (1984) 171–194

Analogy as Exploration

Chris Thornton

COGS/Informatics, University of Sussex, Brighton, BN1 9QH, UK,
c.thornton@sussex.ac.uk, www.christhornton.eu

Abstract. Theorists have often emphasized the role of analogy in creativity, particular in scientific or intellectual contexts. But in Boden's model, creative processes are assumed to be mediated by acts of exploration and transformation. There is a need to understand how these two perspectives fit together and to clarify whether analogy is mediated by transformation, exploration, or both of those processes. Analogy is often viewed as a form of transformation. But the demonstration that transformation can be understood as meta-level exploration calls that view into question. The present paper shows that analogy can in fact be mediated by exploration alone. A task analysis is used to delineate the prototypical strategies of concept construction. It is then shown that analogical functionality can emerge as a natural result of conceptual-space exploration.

Keywords: Analogy, exploration, transformation, creativity, theoretical cognitive science, concepts, representation.

1 Introduction

Many theorists identify analogy as fundamental for creativity, e.g., [1; 2; 3; 4; 5; 6]. However, the transformational model of creativity [7; 8; 9; 10] emphasizes the role of search-like operations applied to concepts. Creativity on this latter view is understood to involve heuristically-guided exploration of existing conceptual spaces and *transformation*, development of new conceptual spaces. The question then arises as to how analogy fits into this scheme. Is it that analogy should be understood as being an ingredient of conceptual space transformation? Or is it better to think of it as a part of exploration? Using a task analysis of conceptualization [11], the present paper shows that analogical functionality is an emergent property of ordinary conceptual-space exploration.

2 Conceptualization prototypes

For purposes of the analysis, it is assumed that exploration of conceptual spaces can be understood to involve the construction of new concepts from given constituents. On the basis that these are themselves concepts (or can be viewed as such) the task can be accomplished in one of two ways. The constituents man be treated as independent entities. Or they may be treated as a related ensemble.

In the former case, the result is necessarily one in which a set of independent entities are treated as a single entity. It is a generalization or abstraction, i.e., a category which includes all the constituents. In the latter case, the nature of the result depends on the relationship which is deemed to tie the constituents together as an ensemble.

On analytic grounds, then, we can divide the task of concept-combination into two basic forms:

- *categorical* construction in which the constituents are deemed to be independent, and
- ensemble or *compositional* construction, in which the constituents are deemed to be tied together by a relationship.

Where a concept-combining agent is able to introduce relationships, then, there are two ways of accessing new concepts. For any given combination, a categorical construct can be formed. The constituents come to be viewed as instances of a category. For the same combination, each way of superimposing a relationship produces a distinct compositional construct. In this case the constituents come to be viewed as part of the superimposed relationship. This provides us with a provisional account of the possibilities of conceptual space exploration.

In order to identify the concepts which can be generated in a particular case, we need to know what concepts are assumed to be given, what relationships are assumed to be given, and how they can be applied. It simplifies matters if we treat given concepts and given relationships the same way, i.e., if we assume that among given concepts, some are relational and some non-relational. Assuming no restrictions on how such relational concepts can be applied (i.e., taking them to be unordered relations of arbitrary arity), the number of concepts which can be formed by combination can then be ascertained straightforwardly. For each combination of given non-relational concepts, there is

- one categorical construct, and
- for each given relational concept, one compositional construct.

Fig. 1 illustrates how things work out when there are three non-relational and two relational constituents. The given concepts are represented here as circles in the bottom row. The non-relational concepts x, y and z are in the middle. The relational concepts r_1 and r_2 are on the outside. Circles in the top row represent the possible combinatorial constructs with the arcs denoting constituency. The three circles on the left are the categorical constructs. In this case, arcs are brought together indicating the way in which the constituents are combined into a single entity by the construction. Next to these we have the three constructs obtained by applying relationship r_1, followed by the three that can be formed by applying r_2. The connecting arcs here connect to a horizontal bar labeled with the relation applied. This indicates the way in which the relational concept is applied to schematize the ensemble's relationship. The figure shows precisely how many conceptual structures can be formed in this scenario. With three

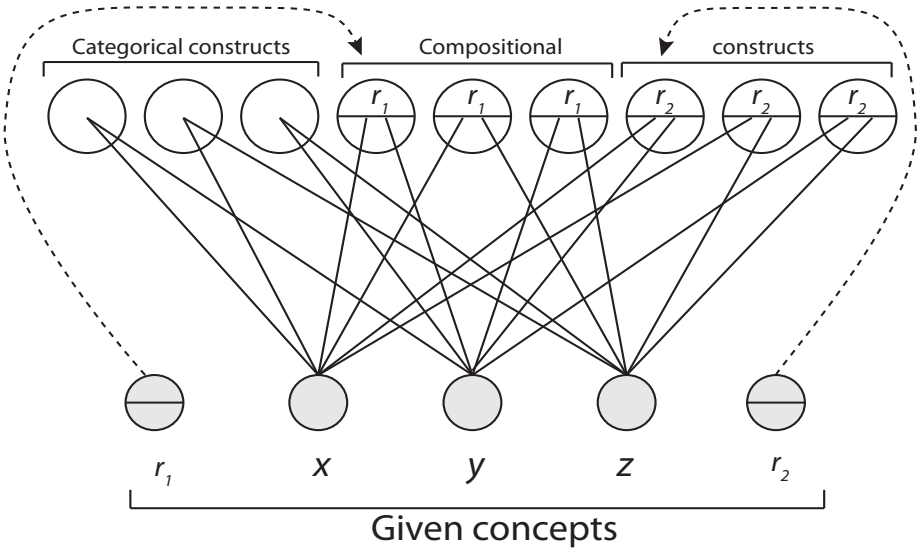


Fig. 1. Categorical and compositional concept-combination.

non-relational and two relational givens, there are nine conceptual structures in all.

3 Hierarchical development

We can map out the contents of the conceptual space defined by some givens, then, by looking at the possibilities for categorical and compositional construction. But Fig. 1 merely shows the concepts which can be obtained directly. We also need to consider the concepts which can be obtained indirectly, i.e., by pursuing constructive possibilities in which previous constructions are used as constituents.

The possibilities are illustrated in Fig. 2. Here, the given and constructed concepts are taken from a domain of vehicles and a domain of animals. However, the concept names should not be taken too seriously: the intention is simply to show syntactic/combinatorial possibilities as before. The given non-relational concepts are BICYCLE, POLICE-CAR, LION, ZEBRA1 etc. The given relational concepts are BETWEEN, STOPPING, CLOSE-TO etc. In this diagram, however, the relational concepts only shown where they are *applied*.

The schematic on the far-left represents the simplest situation. Here the constructed, first-order[1] concept POLICE-ESCORT is used as a constituent in the construction of a 2nd-order compositional construction based on the STOPPING relation. Note that, here, the relation applied at the second level is different to

[1] The term 'order' is used to denote hierarchical level.

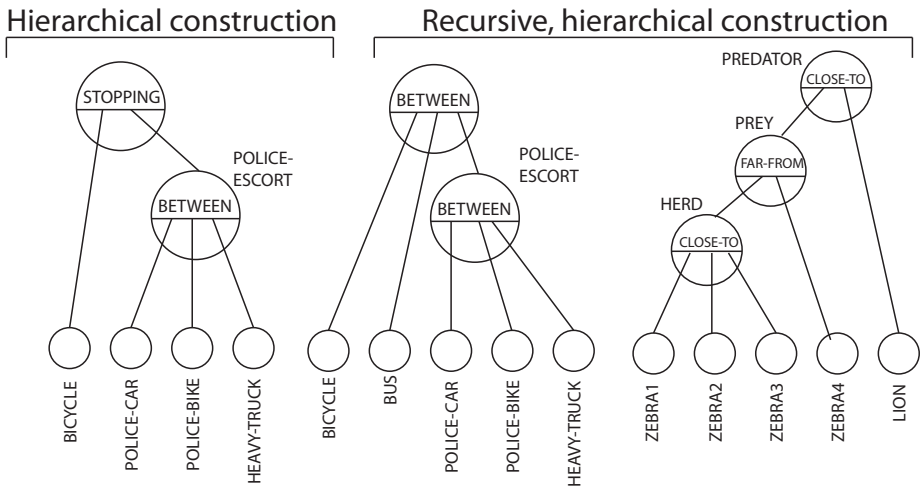


Fig. 2. Hierarchical and recursively hierarchical construction.

the one used at the first level. But re-using a relation is also possible. This leads to the recursive forms of construction illustrated in the middle and right schematics. In the middle schematic, the BETWEEN relation is used to produce a compositional construction on constituents which include one constructed using the same relation. The right schematic also illustrates recursive use of a relation (CLOSE-TO) but here there is an intermediate compositional construct based on the FAR-FROM relation.

4 Analogy as conceptualization

In Gentner's structure-mapping theory [12; 13; 14], analogy is understood to be the exploitation of a mapping between two conceptual structures. The mapping allows details in one structure (the 'target') to be inferred from details in the other (the 'source'). Gentner posits the *abstractness* principle — the notion that the strength of the analogy depends on the degree to which relations rather than attributes are preserved in the mapping. She also proposes the *systematicity principle*, which is the idea that strength also depends on the degree to which 'systems of relations' are carried across from source to target [12, p. 158].

Gentner treats analogy formation as a special operation. However, bringing to bear the analysis of concept combinatorics, we can understand it to be a natural consequence of conceptual-space exploration. What Gentner describes as an 'analogical mapping' can be understood to be a categorical construct combining two compositional constituents, which themselves embody structures of constructs exhibiting corresponding relations.

Figure-shown contrasts a conceptual construction which embodies an 'analogical mapping' with one that does not. In the left schematic, the correspondence

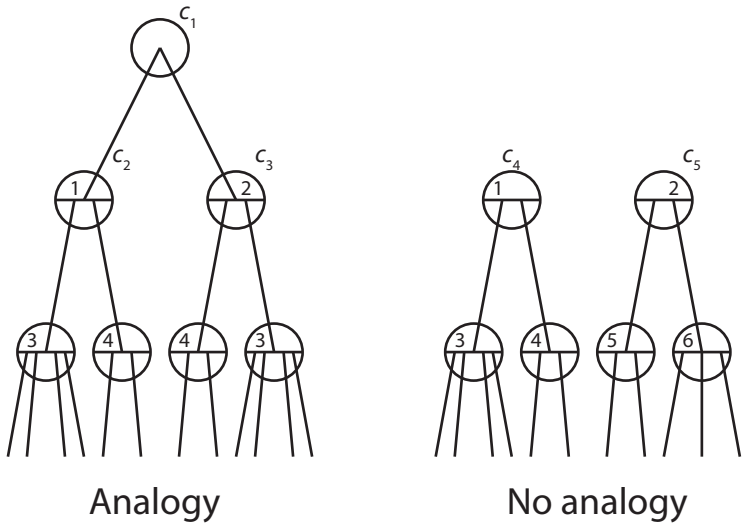


Fig. 3. Formation of analogical mapping through conceptual-space exploration.

between the relations used in the construction of the 1st-order constituents of c_2 and c_3 implies the existence of an analogical mapping between them. The construction of the categorical construct c_1 then places c_2 and c_3 into a single category, implicitly realizing the analogy. In the right schematic, there is no relational corresondence at the level of 1st-order constituents and therefore no analogy.

The formation of such analogical mappings is a natural consequence of conceptual-space exploration. But Gentner also posits abstractness/systematicity, i.e., a *preference* for the formation of such mappings. While this principle is not itself inherent in conceptual-space exploration, it is not hard to envisage how it might emerge from it. For example, a preference for the formation of a categorical construct on c_2 and c_3 (rather than c_4 and c_5) might be based on recognition that, in using a smaller number of relational concepts, the former has a lower representational cost.

5 Analogical transfer

An important feature of Gentner's structure-mapping model is the way in which it accounts for analogical transfer, i.e., the filling-in of relations in the target on the basis of features observed in the source. This is often illustrated using the case of the 'Rutherford' analogy. Fig. 4 is Gentner's own schematic which shows how the structure of relations affecting planetary motion are related to those affecting electrons orbiting a nucleus. On Gentner's analysis, the strength of the analogy depends on the correspondence between the relations (ATTRACTS,

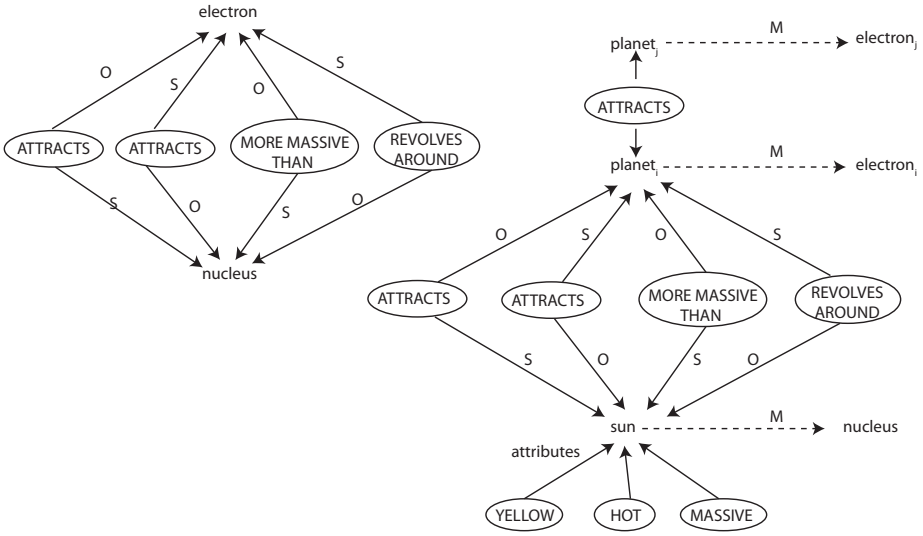


Fig. 4. Gentner's structure-mapping model of the Rutherford analogy.

REVOLVES-AROUND etc.); its formation allows the ATTRACTS relation in the planetary structure to be transferred, effectively filling-in an unknown property of the electron structure.

Reproduction of this functionality in conceptual-space exploration is illustrated by Fig. 5. In forming the top-level, categorical construct c_1, the structures subsumed by c_2 and c_3 are generalized, implicitly filling-in the missing ATTRACTS relation in c_3. Conceptual-space exploration can thus serve to 'transfer' relations subsumed in c_2 to c_3, thus reaping the benefit of the analogical correspondence.

6 Copycat analogy as conceptualization

Gentner's structure-mapping theory has influenced a broad range of work. Analogy models such as the ACME system of Holyoak and Thagard [15] aim to extend Gentner's account by showing how the mapping processes can be implemented or applied. Other models may model analogy without treating SMT as a foundation. An interesting case from the present perspective is the Copycat system of Melanie Mitchell and Douglas Hofstadter, [3; 4]. This is a pseudo-parallel[2] stochastic system which constructs analogical mappings in a data structure called the 'workspace'.

The formation of constructs in Copycat is carried out by 'codelets' (also described as 'micro-processes'). An agenda structure termed the 'coderack' is

[2] In Hofstadter's term, the system executes a 'parallel terraced' scan [3, p. 33].

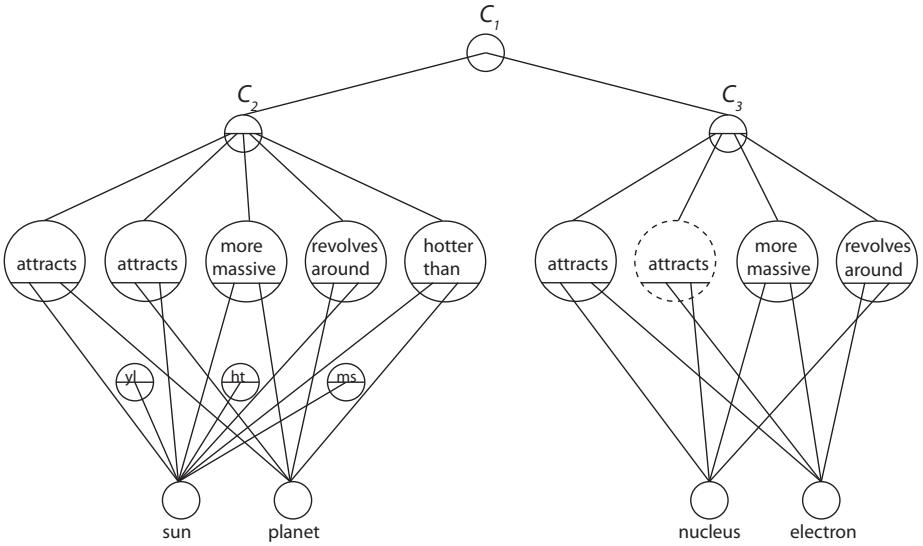


Fig. 5. Transfer of relations in conceptual-space exploration (yl, ht and ms represent the properties of yellow, hot and massive here).

used to mediate the task of scheduling constructive operations. These attempt to achieve 'conceptual slippage' by exploiting relationships in a data structure termed the 'slipnet'. Behind this terminology, however, Copycat pursues a relatively familiar constructive protocol. It builds six types of construct, one being the universal categorical construct *group* and the other five being different types of compositional construct [3, p. 40]. In terms of the present framework, we would say that it utilizes five compositional relations which, figuring in categorical/group construction, allows construction of six types of construct in all.

The constructive processes pursued by Copycat in forming an analogical mapping can also be reproduced by conceptual-space exploration. Consider Fig. 6. This is a screenshot of the final workspace generated by Copycat[3] in solving the letter analogy problem:

if 'abc' goes to 'abd' what does 'ijk' go to?

The way in which the system constructs this analogy involves the construction of various types of linking construct. The most critical of these are represented in the screenshot using directed arcs and labeled boxes. The letter-groups 'abc' and 'ijk' are placed into 'group', 'whole' and 'successor-group' constructs. The final 'd' in 'abd' is placed into a successor relation with the final 'k' in 'ijk'. A construct is then built to represent the higher-order relation in which

[3] The public domain applet version of the system from `http://www2.psy.uq.edu.au/CogPsych/Copycat/` was used to generate this screenshot.

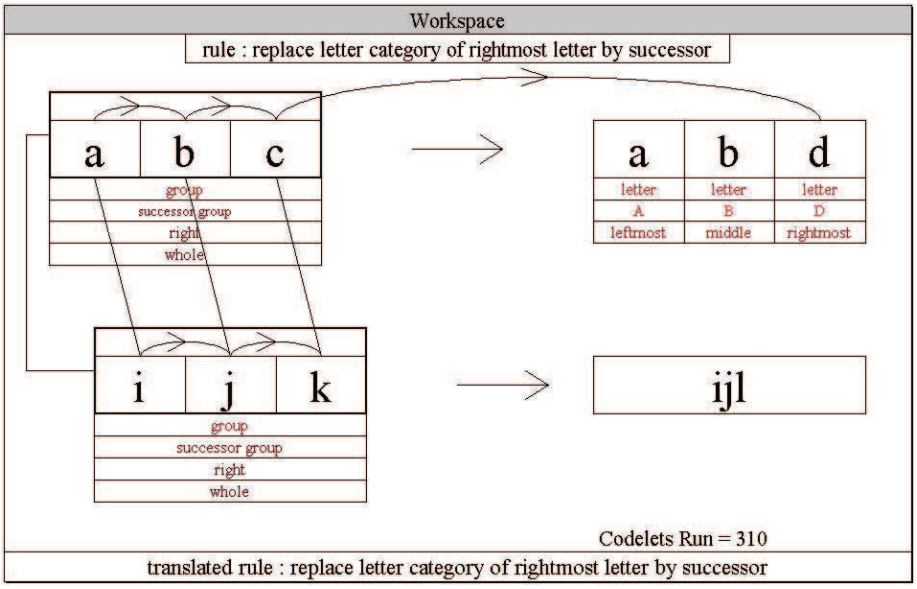


Fig. 6. Construction of a letter-sequence analogy in Copycat.

'abd' is derived by replacing the rightmost letter in 'abc' with its successor. By these means, the system discovers the possibility of there being an analogical relationship between 'abc->abd' and 'ijk' thereby generating 'ijl' as a solution to the problem. (For further details on this and related examples see [3].)

The situation depicted in Fig. 6 is recast as conceptual-space exploration in Fig. 7. Here the critical constructive operations are shown as the formation of categorical and compositional constructs. But the formation of constructs consolidating the analogy (only implicitly represented in the Copycat screenshot) is now explicitly represented. Two, third-order compositional constructs are shown. One of these applies the relation 'succ-group-right-advance' to the relevant constituents which mutually indicate the replacement within a successor-group of the rightmost letter by its successor. The other applies the relation 'whole-right-advance' to the constituents which *would* indicate the (analogous) replacement within a 'whole' gropup of the rightmost letter by its successor. Also shown explicitly is the fourth-order categorical construct which consolidates the root of the analogical match.

7 Concluding comment

While theorists have often viewed analogy as playing a fundamental role in creativity, Boden's model is constructed in terms of operations in/on conceptual spaces. The question then raised is how the process of analogy fits in to Boden's scheme. Boden herself has stressed the way in which the formation of

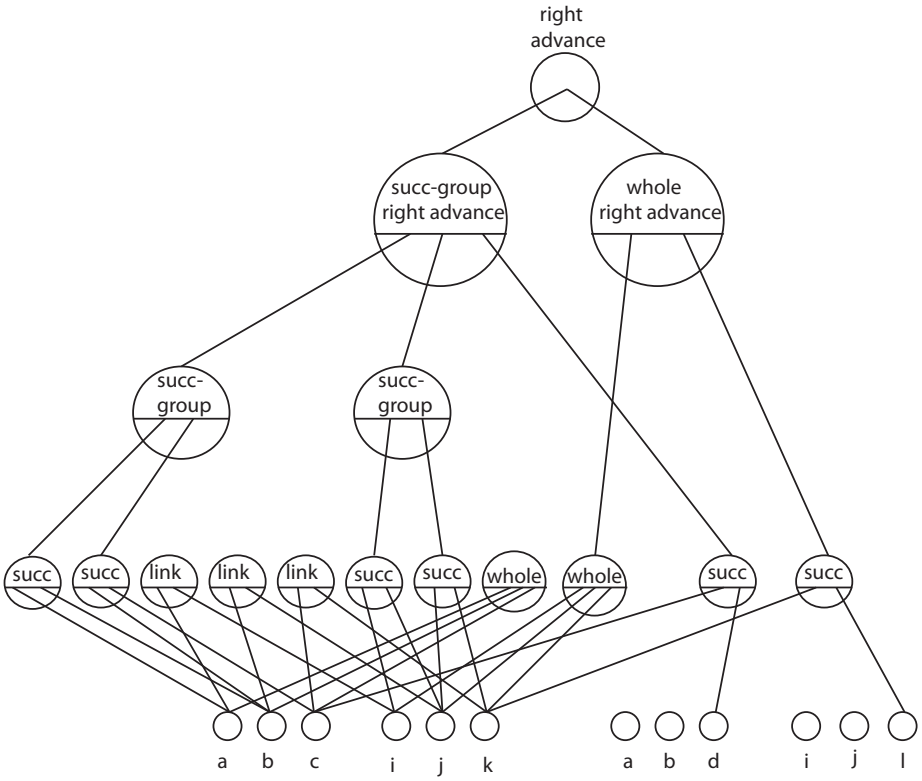


Fig. 7. Copycat construction interpreted as conceptualization.

analogy can be a key element of transformation, i.e., the means by which the constituents of a new conceptual space are formed. This poses some problems in view of the way in which transformation has been shown to be essentially the operation of exploration carried out at the meta-level [16; 10]. However, using an analysis of conceptualization prototypes, the present paper has shown how basic processes of exploration can suffice. Analogy formation may be seen as the formation of a third-order categorical construct built in terms of second-order compositional constructs, which are themselves built in terms of first-order compositional constructs. The potential for analogy-formation is thus inherent in ordinary, explorative functionality.

References

[**1**] Koestler, A. (1964). *The Act of Creation*. London: Hutchinson.
[**2**] Perkins, D. (1981). *The Mind's Best Work*. Cambridge, Mass.: Harvard University Press.

[**3**] Mitchell, M. (1993). *Analogy Making as Perception: A Computer Model.* Cambridge, Mass.: Bradford Books.

[**4**] Hofstadter, D. (1995). *Fluid Concepts and Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought.* Basic Books.

[**5**] Holyoak, K. and Hummel, J. (2002). Understanding analogy within a biological symbol system. In D. Gentner, K.J. Holyoak and B.N. Kokinov (Eds.), *The Analogical Mind: Perspectives from Cognitive Science* (pp. 161-195). London: The MIT Press.

[**6**] Dartnall, T. (2002). Introduction. In T. Dartnall (Ed.), *Creativity, Cognition and Knowledge: An Interaction.* London: Praeger.

[**7**] Boden, M. (1990). *The Creative Mind: Myths and Mechanisms.* London: Weidenfeld and Nicolson.

[**8**] Boden, M. (1998). Creativity and artificial intelligence. *Artificial Intelligence*, *103* (pp. 347-356). 1-2.

[**9**] Boden, M. (2003). *The Creative Mind: Myths and Mechanisms* (2nd edition). London: Weidenfeld and Nicolson.

[**10**] Wiggins, G. (2006). Searching for computational creativity. *New Generation Computing*, *24* (pp. 209-222). Ohmsha, Ltd and Springer.

[**11**] Thornton, C. (2007). How thinking inside the box can become thinking outside the box. *Proceedings of the 4th International Joint Workshop on Computational Creativity.*

[**12**] Gentner, D. (1983). Structure-mapping: a theoretical framework for analogy. *Cognitive Science*, *7* (pp. 155-170).

[**13**] Forbus, K. and Gentner, D. (1986). In R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eds.), *Learning Physical Domains* (pp. 311-348). Los Altos: Morgan Kaufmann.

[**14**] Gentner, D. and Toupin, C. (1986). Systematicity and surface similarity in the development of analogy. *Cognitive Science*, *10* (pp. 277-300).

[**15**] Holyoak, K. and Thagard, P. (1989). A computational model of analogical problem solving. In S. Vosniadou and A. Ortony (Eds.), *Similarity and Analogical Reasoning* (pp. 242-266). New York: Cambridge University Press.

[**16**] Wiggins, G. (2006). A preliminary framework for description, analysis and comparison of creative systems. *Knowledge-Based Systems* (pp. 449-458). Volume 19, Issue 7, November 2006. Elsevier B.V.

A Computational Model for Constructing Novel Associations

Kaz Grace[1], Rob Saunders[1], and John Gero[2]

[1] University of Sydney, Sydney, Australia.
[2] Krasnow Institute for Advanced Study, Virginia, USA and University of Technology, Sydney, Australia.

Abstract. This paper presents a computational model for the construction of novel associations as a component of a larger project on analogy-making. Association-construction is driven by a reinterpretation-based model of subjective similarity. Associations are constructed by transforming the way the agent perceives the objects being associated so that they become similar. The paper describes a model where an agent develops associations by an iterative process of attempting to relate objects in its environment. Associations are constructed by building appropriate transformative interpretations. Methods for the learning of transformations are discussed. The capabilities and implications of the model are discussed through an example application to the domain of geometric proportional analogies.

Keywords: association, novelty, analogy-making

1 Introduction

Computational models of analogy-making have paid significantly more attention to the creation of mappings between existing representations than to the issue of how such compatible representations arose. Reviews of the field of analogy-making [[1], [2]] discuss the need for analogical research to explore the integration of analogy into cognitive processes for the production of relevant, useful and creative analogies. Creative analogies must by definition be based on associations that are novel. This paper presents a computational model for the construction of novel associations as a step towards the computational production of creative analogies.

This research defines an association as an interpretation of two objects which expresses a relationship between them. An analogy is defined as association-based but additionally incorporating the transfer of knowledge from the source to the target driven by the goals of the analogy-making attempt. A critical feature of associations (and by extension analogies) is that they are situated; they exist within a specific way of thinking about both the objects being associated and the world they are in. Associations between objects are highly subjective and dependent on an agent possessing particular representations of those objects.

In this paper we assume that the relationship between two objects is defined by the manner in which those two objects are being interpreted. It follows that the problem of constructing a new relationship between the objects can be modelled as the problem of constructing an interpretation of them such that they can be associated. In other words the critical problem in finding new associations becomes the search for appropriate interpretations rather than the search for possible mappings within interpretations.

A method for describing interpretation as a transformation is described with reference to a simple example in the domain of colour. A computational model for constructing novel associations, the "association game", is developed based on transformational interpretation and described formally. The model uses an iterative, experience-driven process of constructing interpretations of objects in which a new relationship can be perceived. An example association game is presented in the domain of geometric shapes. The place of this model in analogy-making and creativity research is discussed.

2 Interpretation-driven association

An interpretation is a perspective on the meaning of an object; a particular way of "looking" at an object. The process of interpretation involves the construction of that viewpoint. This process is described in this research using the theory of conceptual spaces [[3]], in which concepts exist as regions in a conceptual space. A conceptual space is an abstract construct defined by a number of quality dimensions, or "aspects or qualities of the external world that we can perceive or think about" [[4]], for example 'weight' or 'time'. Quality dimensions have associated spatial structures; weight is one-dimensional with a zero point and is isomorphic to the half-line of non-negative numbers, while hue is circular. These structures impart topological properties on regions; it is meaningful to talk about "opposite" colours but not "opposite" weights.

We represent interpretation as a transformation applied to a conceptual space. As concepts are represented by regions of the conceptual space, the application of a transformation to the space causes different features to be identified with objects within it. A new association is constructed when transformations are applied to one or both objects so that some shared features, that were not present before the transformations were applied, emerge. In transforming the conceptual space as it applies to the objects in the association the agent has created a new interpretation of the objects in which a mapping is possible.

The notion of learning a transformation to solve an association problem can be likened to the approach used in proportional analogy problems of the form $A : B :: C :?$, in which a transformation between A and B must be learned in a way that it can be generalised to C and used to produce D. This research focuses on the construction of the relationship on the left hand side of proportional analogy problems, but differs from previous proportional analogy systems [[5], [6]] in that relationships are constructed rather than selected.

Figure 1 shows a very simple example of a transformative association in the Hue, Saturation, Brightness space of colours. In the unmodified diagram on the left the two colours, while both greenish in hue, are clearly different. In the reinterpreted diagram on the right the source object has been transformed by "swapping" the values in the saturation and brightness dimensions. Depending on the specificity of the concepts possessed by the agent they could now be both identified as "dark green". The interpretation that is being used to construct this association could be phrased in English as "treat saturation as brightness and brightness as saturation and the source and target can be thought of as alike".

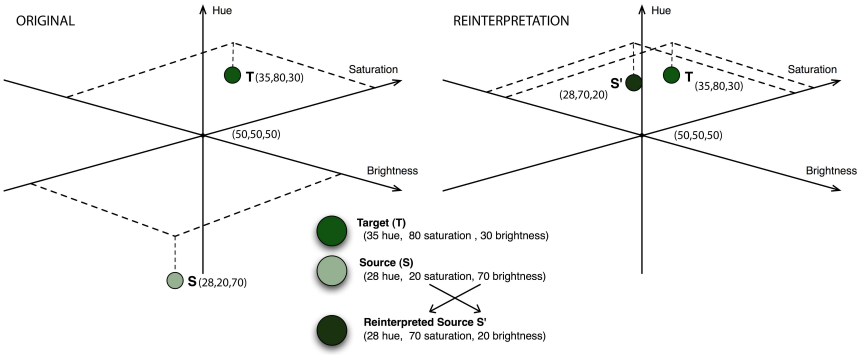


Fig. 1. An example of a transformation that enables an association. The Saturation and Brightness dimensions have been exchanged and the reinterpreted source can be related to the target.

3 The association game

Generating interpretation-based associations is an iterative process that produces multiple "failed" interpretations, where failure is defined as not producing an appropriate mapping or indeed any mapping at all. Learning from these failed associations enables the construction of additional associations driven by past experiences. In this research we model association-construction through the construct of the "association game" an iterative experience-based process of reinterpretation. The game is based on the discrimination agent architecture developed by Steels [[7]]. Steels developed a way for agents to learn a grounded model of their world in the course of attempting to distinguish objects from their surroundings. Agents played a series of "discrimination games" in which they constructed feature detectors that were suited to identifying an object from its context. The agent constructs new feature detectors or refines existing ones as

necessary to distinguish between objects. Through repeatedly learning to identify each object from its context the discrimination agent learns about all the objects in its world.

This model adapts the discrimination agent architecture for an alternative purpose: learning associations grounded in use. In this model an agent attempts to learn about its world by creating new ways to relate objects. The agent initially attempts to discriminate between objects, but if it is capable of telling two objects apart it will attempt to reinterpret them in such a way that a new relationship between them is created. The interpretation that enabled the new association persists the next time a game is played between the same two objects and as a result the discrimination phase of that next game will differ. Successive games played with the same objects will lead to sets of associations composing a coherent mapping between the source and target.

As multiple games occur with the same objects and new interpretations supersede old, the agent must evaluate whether the mappings enabled by the new interpretation warrant keeping. As the current model does not incorporate analogical transfer this evaluation cannot be performed based on the reason the agent is making the association. An association is kept if it is more valuable than the existing mappings (if any) according to evaluation criteria specific to the domain and problem. A preliminary model of this evaluation process could be based on the number of features mapped.

The model as described here focuses on the relationships that can be constructed between objects. The present model requires that both a source and a target object are specified. Associations are generated between the objects and over time the agent develops associations for multiple pairs in its environment. The process of interpretation used by the system in constructing associations is situated in that interpretations are driven by a world view that is constructed from previous experiences. The agent attempts to reinterpret the source in ways that have worked in the past for similar problems. For each association game the agent's memory constructs associations that are believed to be similar to the current problem and attempts to apply them. The features that the agent learns to detect are modelled as regions in the agents conceptual space and an association is modelled as a transformation within that space that produces an overlap between the features of two objects.

3.1 Formal description

An agent a exists in a world containing a set of objects $O = \{o_1, \ldots, o_n\}$. The agent possesses a set of sensory channels $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$ which are functions over O. Each sensory channel σ_j defines a value for each object o_i. An agent has a set of feature detectors $S_a = (s_{a,1}, ..., s_{a,p})$ which is initially empty. A feature detector $s_{a,k}$ consists of a function $\phi_{a,k}$ over a sensory channel σ_j that is updated after each game. The feature set derived by applying the feature detectors of an agent a to an object o_i is defined as F_{a,o_i}. A distinctive feature set D^C_{a,o_i} is a set of features that serves to uniquely identify an object o_i to agent a from a context of other objects $C \subseteq O$. For formal definitions of F_{a,o_i} and D^C_{a,o_i} see

Steels [[7]]. In this model the context C of an association game is always of order two, containing a source o_s and a target o_t.

An interpretation T_{a,o_i} is defined as a transformation applied to the feature detectors of agent a when they are applied to an object o_i, denoted $F_{a,T(o_i)}$. An association $\alpha(o_t, T(o_s))$ is a relationship between a source object o_s and a target object o_t such that when the interpretation T is applied to the source there are new common features detected in o_t and o_s. In other words $\alpha(o_t, T(o_s))$ implies that the set of mapped features $K \neq \emptyset$, where $K = \{f \mid f \in (F_{a,o_t} \cap F_{a,T(o_s)}), f \notin (F_{a,o_t} \cap F_{a,o_s})\}$.

E_a is the set of all experiences possessed by agent a. An experience $e \in E_a$ is a memory of an association $\alpha(o_t, T(o_s))$, containing distinguishing and common features of the source and target in addition to the transformation T. A situation $X_{a,g}$ is constructed from a set of experiences that are applicable to the current game g, defined as being within a threshold d given a similarity metric $\Delta_{X_{a,g}}$.

The association game $g = (a, o_s, o_t, \alpha)$, where a is an agent, o_s and o_t are objects chosen from O and α is the association that has been developed for this pair of objects (if any), proceeds as follows:

1. The agent determines the distinctive feature sets D^C_{a,o_t} and $D^C_{a,T(o_s)}$, where T is the interpretative transformation used in α (if any). For a more in depth discussion of selection criteria for distinctive sets see Steels [[7]].
2. The distinctive feature sets and the common feature set $(F_{a,ot} \cap F_{a,T(o_s)})$ are used as cues to construct a situation $X_{a,g}$ from E_a. The transformation T' is constructed from experiences in E_a as applied to the current objects using $X_{a,g}$.
3. The agent determines whether to keep the new interpretation based on whether the application of the transformation T' yields more valuable mappable features than the previous transformation T. In principle the evaluation metric is determined by the purpose of the analogy, but a proof of concept implementation could be based on simple enumeration of mapped features.
4. If the reinterpretation using T' is successful in creating a new mapping the agent creates a new association $\alpha'(o_t, T''(o_s))$ where T'' is the new transformation T' applied to the existing transformation T. A new experience e' is added to E_a based on α'.

The association game can end in failure if one of the following conditions occurs:

1. $D^C_{a,o} = \emptyset, o \in C$. There are not enough distinctions in S_a to identify either o_t or o_s and therefore $\forall o_c \in C, F_{a_r}, o \subseteq F_{a_r,o_c}$. When this occurs, the agent will construct a new feature detector or modify an existing one, see Steels [[7]] for the specifics of this process.
2. $X_a = \emptyset$. The agent has no experience applicable to the current context. In this case the agent analytically calculates a simple transformation of the source that will produce commonality between its distinctive features and those of the target. This transformation produces a mapping of the discriminating features of two objects between which no mapping previously existed.

3. $order(K') < order(K)$. The transformation that was applied results in a worse mapping than the one it supersedes. In this case the new transformation T' is discarded and T remains as the association in α. Currently no experiences are generated on a failure case.

4 An example association

The association game model can in theory be applied to any domain for which an appropriate conceptual space can be constructed. To illustrate the game process we present a simple example in the domain of geometric proportional analogies. Proportional analogy problems of the form $A : B :: C :?$ are suitable for transformation-based association because the relationships A to B and $A : B$ to $C :?$ can be easily understood in terms of transformations. This model formalises the transformations by applying them at the level of conceptual spaces rather than to the problem directly. The example focuses on the association between two objects, or to put it in terms of a proportional analogy; constructs the relationship $A : B$.

An example proportional analogy problem can be seen in Fig. 2. In this example we assume that the agent has already learnt concepts suitable to describe and discriminate between the two figures A (the source) and B (the target). The conceptual dimensions used in this example, such as the recognition of attributes of sub-shapes within the figures, are for illustrative purposes only. Perceptual processes in an implementation of this model may produce different conceptual spaces.

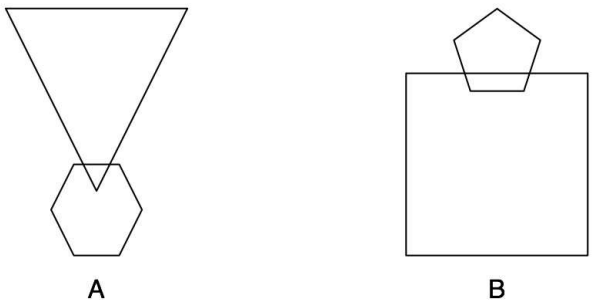


Fig. 2. A pair of figures as they are perceived by an agent playing an association game. Each figure has four features relevant to this example: the size and side counts for both the upper and lower shapes.

The game proceeds through the four steps outlined in Section 3.1. For ease of explanation we assume that the agent has encountered sufficient objects similar to these to possess the necessary conceptual lexicon. We also assume a very

simple conceptual space for this example: a four dimensional space for each figure; the size and number of sides of both shape elements (starting with the upper shape), expressed ($size_1, sides_1, size_2, sides_2$). Many other possible spaces exist for this and other proportional analogy problems, this space is not an example of the spaces that might be used by the agents but an explanatory aid.

The first step is to describe and discriminate between the two objects. The agent's feature detectors identify object A as having a top shape of size 4 with three sides, and a bottom shape of size 2 with six sides, denoted in this conceptual space as (4,3,2,6). Object B is identified as having a top shape of size 2 with five sides, and a bottom shape of size 4 with four sides, denoted (2,5,4,4). There are several possible distinctive feature sets between A and B, but we will assume that the agent discriminates based on the size of the upper shape - the agent states that the upper shape is "large" in figure A and "small" in figure B.

The second step is to try reinterpreting the source, A, by constructing an appropriate transformation that the agent believes will lead to a new relationship with B. Based on the feature sets and on the agent's prior experience with similar problems one possible transformation would be to treat the sizes of the two shapes within the figure as if they were reversed. The interpretation here would be "treat the size of the upper shape as if it were the size of the lower and visa versa". The problem can be seen in Figure 3 with the transformation applied.

The third step in the association game is to compare the reinterpreted source object with the target to determine In conceptual space the new co-ordinates of the source A would be $(2, 3, 4, 6)$. When this is compared to object B a new association could be constructed as the agent's feature detectors would now identify the agents by the same features on both the $size_1$ and $size_2$ dimensions. In other words when the agent "thinks of" figure A in this new way it is somehow "the same" as figure B - a relationship has been constructed where none was present before. Previously there was no association, so the agent views the re-interpretation as a success.

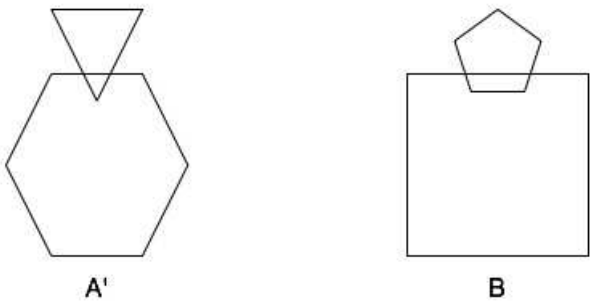


Fig. 3. The figures from Figure 2 after A has been reinterpreted to produce A'. A' is how the agent perceives A after its conceptual space has been transformed. An association with B can now be constructed.

The last step in the association is to store the new association as an experience. This directly affects future association games between A and B as the stored transformation will be already applied and the new game will pick up from where this one left off. The experience also affects games between other objects as it can be used in the construction of other transformations. As a result of this experience and other association games the agent may learn to attempt to exchange dimensions between shapes within a figure.

A possibility in a future game applied to this example is that the agent attempts to create a mapping between the $sides_1$ and $sides_2$ dimensions. In other words the agent may apply its experience in swapping the sizes of the upper and lower shapes to the side counts. The number of sides is not an exact match as the shapes sizes were, with three and six sides for the shapes in figure A and five and four for the shapes in figure B. As a result of this the second mapping would depend on the specificity of the feature detectors that have developed for the number-of-sides dimensions. If the agent is specific enough to differentiate between 3 and 4 or 5 and 6 sided objects, then the second transformation would need to be more complex than a simple dimensional exchange for the reinterpreted source to be seen as "the same" in these dimensions. An informal English translation of the reinterpretation of shape A created by the application of both transformations would be "Treat the upper shape as if it were the lower and the lower shape as if it were the upper". The results of this second transformation can be seen in Figure 4.

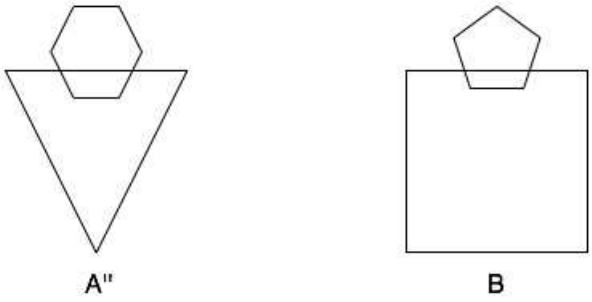


Fig. 4. The figures from Figure 2 after A has been reinterpreted twice to produce A". A" is produced by applying the same kind of transformation that produced A' to different features.

5 Discussion

The model presented in this paper describes a method of constructing novel associations in which mappings are built over repeated iterations of the associ-

ation game. The process of association is based on transformations applied to the conceptual space from which feature-based representations are generated. The model addresses the problem of representational rigidity suffered by symbolic association-based systems [[8]] by basing mappings on reinterpretations rather than symbolic identity. Through reinterpreting the source object the system is able to construct mappings between features of the objects that do not have prior symbolic connections. This ability is mediated by an evaluation system that discards interpretations that enable the mapping of no additional useful features. Through iterative performance of the experience-driven association game the agent constructs an interpretation of the source object that enables mappings between those features that are appropriate within the interpretation. The association that is constructed between the source and target represents a relationship between them that is novel.

The agent playing association games learns about different ways of reinterpreting objects by learning different kinds of transformation and problems to which they can be applied. The agent learns transformations that are grounded in the specific set of objects it is exposed to and the sensory channels it possesses. In this paper we make the assumption that the learning system is capable of storing and applying transformations that have been useful to similar stuations where they may be similarly useful. We also assume that sufficient regularity exists in the environment that such appropriate transformations can be learnt and generalised to multiple associations in the domain. These assumptions are important points to be addressed in the development of systems that implement this model.

The system currently relies on its ability to discard superficial mappings to ensure that appropriate matches can be found and others eventually discarded. This process could be improved by more tightly integrating the discrimination phase of the game with the goals of the association process. If the association was being made for a purpose the discriminating features could be selected based on their applicability to that purpose. The evaluation of whether a new interpretation is successful could also be integrated with the goal of the association, producing associations that may be useful to the agent in addition to being novel.

The model presented in this paper is a process for generating relationships that are not based on symbolic identity in static representations but are novel associations constructed through reinterpretation in a situated agent. The construction of associations that are novel to the agents that create them is a critical component of computational analogy-making and of computational creativity.

References

1. French, R.: The computational modelling of analogy-making, *Trends in Computer Science*, (2002) **6**: 200–205.
2. Kokinov, B.: Analogy is like cognition: dynamic, emergent and context sensitive, *Advances in Analogy Research* NBU Press, Sofia, (1998).
3. Gärdenfors, P.: *Conceptual Spaces: The geometry of thought*, MIT Press, (2000).

4. Gärdenfors, P.: Induction, conceptual spaces and AI, *Philosophy of Science*, (1990) **57**: 78–95.
5. Evans, T.: A heuristic program to solve geometric-analogy problems, in *Proceedings of the 25th Spring Joint Computer Conference*. Morgan Kaufman, (1964) pp. 327–338.
6. Mullally, E.C. and O'Donoghue, D.P.: Spatial inference with geometric proportional analogies, *Artificial Intelligence Review*, (2006) **26**: 129–140.
7. Steels, L.: Perceptually grounded meaning creation, in *Proceedings of the Second International Conference on Mullti-agent Systems (ICMAS096)* Los Alamitos, CA, (1996) pp 338–344.
8. Chalmers, D., French, R. and Hofstadter, D.: High-level perception, representation and analogy: a critique of artificial intelligence methodology, *Journal of Experimental and Theoretical Artificial Intelligence*, (1994) **4**: 185–211.

Slip-Sliding Along in Linguistic Creativity: Building A Fluid Space for Connecting Disparate Ideas

Tony Veale[1], Yanfen Hao[1],

[1] School of Computer Science and Informatics, University College Dublin, Dublin D4, Ireland.
{Tony.Veale, Yanfen.Hao}@UCD.ie

Abstract. The art of linguistic creativity lies not in finding new truths to express in language (if there are any), but in finding new and more resonant ways to express truths that are already known or implicitly accepted. Creative expression thus requires that we take adopt a new and revealing perspective on a familiar idea, one that may prompt an audience to conceptually re-represent this idea to draw out new meaning or new relevance to a situation. As such, a computational model of linguistic creativity requires a knowledge representation that is as semantically agile and accommodating as the creative speakers who use it. We thus present a flexible and very concise knowledge representation for dealing with creative metaphors, called *Talking Points*, and show how talking points can be acquired on a large scale both from WordNet and from the web to form a fluid connected structure called a *slipnet*.

Keywords: linguistic creativity, metaphor, humour, re-expression, slippage.

1 Introduction

Linguistic creativity manifests itself in two key ways in language. In one guise, it makes the unfamiliar and the strange seem more familiar and understandable [1]. For instance, one might describe a burqa (a full body covering for Muslim women) as a suit of armor, as a shield against prying eyes or, depending on one's communication goal, as a wearable cage. In its other guise, the one most often associated with the poetic and fanciful use of language, it makes the familiar and mundane seem strange and unfamiliar, allowing us to view a commonplace idea from a new and revealing category perspective. For instance, one might describe make-up as "the Western burqa", to communicate not just the idea that each involves a covering of the female form, but that each reflects a society-imposed expectation on the public presentation of women. Each of these roles is a manifestation of the same underlying mechanism for combining concepts, for understanding how they interact [3] and for determining how they are connected [4], even if those connections are tenuous, hidden or not always obvious [5]. For instance, Burqa and Make-up are connected by a conceptual path that nudges the meaning "something that must be worn by Muslim women" into "something that must be worn by women" into "something that is conventionally worn by women" into "something that society expects women to wear".

Creative linguistic devices like metaphor allow a speaker to draw out and highlight, in a modified or exaggerated form, surprising connections between different concepts. A flexible knowledge representation is thus needed if a computational system is to identify, follow and reconcile these connections. In fact, the kind of fluid representation that is required has already been described by Hofstadter in [6], who emphasizes the role of slippage in semantic representation when dealing with creative phenomena such as formal analogies, linguistic parlor games and other playful uses of language. Such a fluid knowledge representation will define the search space in which creative language processes like metaphor generation and metaphor understanding can be cognitively and computationally situated [7]: for generation, fluid connectivity will allow a system to search outwards from a given target to find those source concepts that offer the newest yet most appropriate perspectives; for understanding, fluid connectivity will allow a system to reconcile those aspects of a source concept that are most relevant to the given target concept.

In this paper we describe the construction of a fluid knowledge representation for creative metaphor processing, one that is acquired automatically from WordNet [1] and from the texts of the web. In section 2 we summarize related work in the field of metaphor as it pertains to flexible knowledge representation. In section 3 we describe two complementary means of acquiring the basic elements of this representation, from WordNet and from the web, before describing how these elements can be placed into a fluid network of connections – what Hofstadter [6] calls a *slipnet* – in section 4. We then present in section 5 some empirical evaluation of the acquired representation on an objective test of term categorization, before concluding with some consideration of future work in section 6.

2 Related Work

Computational work in linguistic creativity has focused on the vexing problem of metaphor interpretation for two reasons: poetic metaphor is a creative phenomenon *par excellence* [3], while conventional metaphor is pervasive is everyday language and an important challenge for NLP [8,9,10,11]. Most approaches embody a sense of what it means to be literal, and accommodate metaphoric meanings within this conventional scheme through a form of relaxation, mapping or translation. Wilks [8] advocates that the *hard* constraints that define a literal semantics should instead be modeled as *soft* preferences that can accommodate the violations that arise in metaphoric utterances, while Fass [9] shows how these violations can be repaired to thus capture the literal intent behind each metaphor. The *Midas* system of [10] explicitly encodes schematic knowledge about conventionalized metaphors such as "to kill a process" and "to open a program", and uses this knowledge to fit novel variations of these metaphors into the most apt schemas. The role of inference in metaphor understanding is instead emphasized by [11], who describe a system called *ATTMeta* that contains sufficient knowledge about e.g., conventional metaphors of mind to support complex inferences about the mental states implied by metaphors.

Hofstadter [6] considers a more formal and tightly-controlled kind of language creativity in the guise of abstract analogies. This toy-like format, which supports tasks

as diverse as the mapping of letter sequences or the mirroring of actions in a highly stylized tabletop environment, allows these authors to focus on the slippage processes that are required to understand analogies whose interpretation is shaped by a wide range of pragmatic pressures. These pressures are modeled using a *slipnet*, a probabilistic network in which concepts are linked to others into which they can easily be transformed or be substituted with. In this view, deeply embedded concepts that are further removed from direct observation are less likely to engage in slippage than more superficial concepts. To take a linguistic example, the choice of word forms in a sentence is more susceptible to slippage (as influenced by e.g., synonym availability in [2]) than the choice of word meanings for that sentence.

Slippage can be seen as a potentially lossy form of conceptual re-representation: the greater the slippage, the more dramatic the re-representation and the greater the potential for loss of accuracy. For instance, a recent magazine cover proclaims the governor of California, Arnold Schwarzenegger, as "president of 12% of the United States". This labeling is creative enough to grace a magazine cover because it involves an ambitious level of re-conceptualization, at least from a computational perspective. The pivotal insights are *Governor* ≈ *President* and *California* ≈ *12% of the United States*. WordNet can be a rich source of insights like the former (since both presidents and governors are characterized as leaders in WordNet), but the latter is an entirely ad-hoc equivalence that one is unlikely to find in any general-purpose resource like WordNet. While ultimately aiming for this kind of creative transformation, our goal here is more modest: to build a network of concepts that are connected by incremental degrees of slippage along pathways of related facts and beliefs. We show how this network can combine the principled flexibility of a Hofstadter-style slipnet with the comprehensive scale of a resource like WordNet.

3 A Knowledge-base of *Talking Points*

We refer to the knowledge elements connected by this slipnet as conceptual *talking points*. We first describe the form of these talking points and how they are acquired, before describing in section 4 how slippage operates between these talking points. We discuss two complementary kinds of talking point here: objective descriptions, extracted from WordNet glosses, and informal, stereotypical descriptions, harvested from the text of the web via a search engine like Google.

3.1 Acquiring Objective Talking Points from WordNet

Objective talking points are aspects of conceptual description that contribute to the consensus definitional view of a concept. Though WordNet does not provide explicit semantic criteria for the definition of each lexical concept, many of these criteria can be gleaned from a shallow parse of the pithy dictionary gloss it associates with each. Thus, whenever the head phrase of a concept's gloss has the form "ADJ$^+$ NOUN" where NOUN can denote a hypernym of the concept, we can associate the talking point is_*ADJ:NOUN* with that concept. For example, the gloss of {Hamas} is "*a*

militant Islamic fundamentalist political movement that …",whichyields the talking points *is_militant:movement*, *is_islamic:movement, is_fundamentalist:movement* and *is_political:movement* for Hamas. When a WordNet concept has a hypernym of the form {ADJ_NOUN}, where NOUN can denote a hypernym of this concept, we likewise associate the talking point *is_ADJ:NOUN* with that concept. For example, {Taliban, Taleban} has {religious_movement} as a hypernym, which yields *is_religious:movement* as a talking point for Taliban.

Objective talking points can also be gleaned from the subject-verb-object structure of a WordNet gloss. For instance, the gloss for synset {conductor, music_director} is "the person who leads a musical group", which yields the talking point *leads:musical_group*. The hypernym of this concept, {musician}, has the gloss "*artist who composes or conducts music …*", which yields the talking points *composes:music* and *conducts: music* that are then inherited by {conductor, …} and other sub-types of musician in WordNet. A shallow parse will generally not lead to a complete understanding of a concept, but will typically produce some interesting talking points of the *predicate:object* variety that can be used to relate a concept to others that are analogically or metaphorically similar. Using WordNet's noun and verb taxonomies, we can identify the following slippage paths between talking points.

composes:music → composes:speech → writes:speech → writes:oration → writes:sermon → writes:law → writes:philosophy → writes:theorem → writes:plan → …

In all, we extract talking points of the form *is_adj:noun* for over 40,000 WordNet concepts, and talking points of the form *verb:noun* for over 50,000 concepts. However, the real power of talking points emerges when they are connected to form a slipnet, as we discuss in section 4.

3.2 Harvesting Stereotypical Talking Points from the Web

The talking points we harvest from the web do not have the authoritative, definitional character we find in hand-crafted resources like WordNet, but they do reflect how people typically speak of (and, perhaps, actually think of) the world. It has been argued in [12] that similes present the clearest window into the stereotypical talking points that underpin everyday conversations, and collect from the web instances of the pattern "*as ADJ as a* *" for thousands of WordNet adjectives. Though the simile frame is shown to be somewhat leaky in English, and prone to subversion by irony, the authors of [12] construct a comprehensive database of more than 12,000 highly stereotypical adjective:noun associations, such as *precise:surgeon*, *straight:arrow*, *balanced:pyramid* and *sharp: knife*. We use their data here, as the basis of an additional web harvesting process to gather stereotypical talking points of the form *has_ADJ:facet*. For every stereotypical association ADJ:NOUN in their database, we send the query "*the ADJ * of a|an|the NOUN*" to Google and collect noun values for the wildcard * from the first 200 hits returned for each query.

This pattern allows us to determine the conceptual attributes that are implicit in each stereotypical *adjective:noun* pairing. For instance, "the delicate hands of a surgeon" and "the inspiring voice of a preacher" reveal that *hand* is a salient attribute

of surgeons while *voice* is a salient attribute of preachers. The frequency with which we find these attributes on the web also allows us to build a textured representation for each concept. So while these expanded web patterns also reveal that surgeons have a thorough *eye* and steady *nerves*, "the hands of a surgeon" are mentioned far more frequently and are thus far more salient to our understanding of surgeons. To avoid noise, the set of allowable attribute nouns, such as *hands, soul, heart, voice*, etc., is limited to the nouns in WordNet that denote a kind of trait, body part, quality, activity, ability or faculty. This allows us to acquire meaningful talking points like *has_magical:skill* for Wizard, *has_brave:spirit* for Lion and *has_enduring:beauty* for Diamond, while avoiding dubious or misleading talking points like *has_proud:owner* for Peacock that lack either representational value or insight. In all, this process acquires 18,794 stereotypical talking points for 2032 different WordNet noun senses, for an average of 9 facet:feature pairs per sense. Specific senses are identified automatically, by exploiting WordNet's network of hypernymy and synonymy relations to connect talking points that describe variations of the same concept

4 Building a Slipnet of Talking Points

To construct a slipnet in the style of [6], but on the scale of [2], we need to connect those talking points that express similar but different meanings, and to quantify the difference between these meanings. Issues of scale mean that we need only connect talking points that are close in meaning, since greater slippage can be achieved by following longer paths through the slipnet. This slippage can be based on semantic *or* pragmatic criteria. Thus, the talking points *has_sacred:authority* (for Pope) and *has_sacred:power* (for God) are semantically similar since the potency sense of "authority" is a specialization of the control sense of "power" in WordNet. Likewise, *writes:speech* and *composes:speech* are similar because "compose" and "write" are synonymous in the context of literary creation, and it is this particular linkage that supports a slippage pathway from *composes:music* to *writes:poetry*. In contrast, *is_political:movement* (for Hamas) and *is_religious:movement* (for Taliban) are pragmatically similar since movements that are religious often have a political agenda also. We can use WordNet to construct the semantic links of the slipnet, but pragmatic links like these require not just word senses but a sense of the world, of a kind we can distil from the text of the web.

Two talking points $is\_ADJ_1:OBJ_1$ and $is\_ADJ_2:OBJ_2$ should be connected in the slipnet if: OBJ_1 and OBJ_2 are semantically close (i.e., synonymous, or semantic siblings in WordNet); and ADJ_1 and ADJ_2 are synonymous, or ADJ_1 frequently implies ADJ_2 or ADJ_2 frequently implies ADJ_1. These implications are recognized and quantified using another web trawling process, in which the query "*as * and * as*" is used to harvest pairs of adjectives that are seen to mutually reinforce each other in web comparisons. This search reveals that "religious" reinforces "superstitious" (5 times), "moral" (4), "political" (3), "conservative" (3), "intolerant" (2) and "irrational" (1). These slippage connections link *is_religious:movement* to *is_political:movement* (a pragmatic shift) to *is_political:campaign* (a semantic shift)

to *is_military:campaign* (another pragmatic shift), thereby connecting Taliban (*is_religious:movement*) to Crusade (*is_military:campaign*).

4.1 Creative Slippage in Action

Slippage is a phenomenon best explained with an example, so consider again the task of creating metaphors for the concept Pope. We have already seen that slippage among talking points allows Pope to be linked to the concept God via Pope→*has_sacred:authority*→*has_sacred:power*←God. Pope can also be linked to Rabbi via the path Pope→*has_sacred:words*→*has_wise:words*←Rabbi and to Judge by the path: Pope→*has_sacred:words*→*has_wise:words*→*has_solemn:words*←Judge. The concept-sensitive interplay predicted by Black's "interaction view" of metaphor [3] is clearly on display here, since the interpretation of a particular source concept depends crucially on how it is able to interact with a specific target concept. The Pope can be metaphorically viewed as a warrior not by considering what it means for a generic person to be a warrior, but by considering how the concept Pope interacts with the concept Warrior, e.g., Pope→*has_infallible:voice*→*has_powerful:voice*←Warrior.

Consider the potential for slippage between objective talking points in WordNet:

Pope ⇒	**Pope** ⇒
≡ *leads:Roman_Catholic_Church*	≡ *leads:Roman_Catholic_Church*
≈ *leads:congregation*	≈ *leads:congregation*
≈ *leads:flock*	≈ *leads:political_movement*
≈ *leads:mob*	≈ *leads:gang*
≈ *leads:organized_crime*	≈ *leads:military_force*
Don (Crime Father) ⇐	**Warlord** (Military Leader)⇐

One can typically terminate a slippage path at any point, to produce different metaphors with varying semantic similarity to the starting concept. Thus, at *leads:flock* one can reach Shepherd, and from *leads:political_movement*, one can reach Civil_rights_leader. A lexicon alone, like WordNet, is generally insufficient for creative metaphors, but such a resource can still reveal useful lexical resonances that may enrich an interpretation. In the example above, we see a resonance between the Pope, which WordNet also lexicalizes as "holy father", and a mafia Don, which WordNet also lexicalizes as "father". Indeed, since WordNet taxonomically organizes {Roman_Catholic_Church} as a specialization of {Organized_religion}, the metaphor creatively establishes a parallelism between crime and religion as organized activities.

5 Empirical Evaluation

To understand whether talking points are sufficiently descriptive of the concepts they are acquired for, we replicate here the clustering experiments of Almuhareb and Poesio [13,14] which are designed to measure the effectiveness of web-acquired conceptual descriptions. These authors use WordNet as a semantic gold-standard, so it

would be circular to replicate their experiments on talking points that are extracted from WordNet. We consider here just the effectiveness of stereotypical talking points.

Almuhareb and Poesio describe two different clustering experiments. In the first [13], they choose 214 English nouns from 13 of WordNet's upper-level semantic categories, and proceed to harvest property values for these concepts from the web using the pattern "*a|an|the * C is|was*". This pattern yields a combined total of 51,045 values for all 214 nouns; these values are primarily adjectives, such as *hot, black*, etc., but noun-modifiers of C are also allowed, such as *fruit* for cake. They also harvest 8934 attribute nouns, such as *temperature* and *color*, using the query pattern "*the * of the C is|was*". These values and attributes are then used as the basis of a clustering algorithm to partition the 214 nouns back into their original 13 categories. Comparing these clusters with the original WordNet-based groupings, [13] report a cluster accuracy of 71.96% using just values like *hot* (all 51,045), an accuracy of 64.02% using just attributes like *color* (all 8934), and 85.5% for both combined (59979 total).

In a second, larger experiment, Almuhareb and Poesio [14] select 402 nouns from 21 different semantic classes in WordNet, and proceed to harvest 94,989 property values (again mostly adjectives) and 24,178 attribute nouns from the web using the same retrieval patterns. They then apply the *repeated bisections clustering* algorithm to this data set, and report an initial cluster purity measure of 56.7% using only property values like *hot*, 65.7% using only attributes like *color*, and 67.7% for both combined. Suspecting that noisy features contribute to the perceived drop in performance, those authors then applied a variety of noise filters to reduce the value set to just 51,345 values and the attribute set to just 12,345 attributes, for a size reduction of about 50%. This leads to an improved cluster purity measure of 62.7% using property values only and 70.9% using attributes only. However, this filtering reduces the clustering performance of both attributes and values combined, to 66.4%.

We replicate here both of these experiments using the same data-sets of 214 and 402 nouns. For fairness, we collect *raw* descriptions for each of these nouns directly from the web, and use no filtering (manual or otherwise) to remove poor or ill-formed descriptions. We thus use the pattern "*as * as a|an|the C*" to collect 2209 raw adjectival values for the 214 nouns of experiment 1, and 5547 raw adjectival values for the 402 nouns of experiment 2. We then use the pattern "*the ADJ * of a|an|the C*" to collect 4974 attributes for the 214 nouns of experiment 1, and 3952 attributes for the 402 nouns of experiment 2; in each case, ADJ is bound to the raw adjectival values that were acquired using "*as * as a|an|the C*". The combination of attributes and values yields a clustering accuracy of 90.2% for experiment 1 (compare with the 85.5% reported in [13]) and an accuracy of 69.85% for experiment 2 (compare with the 66.4% reported in [14]). Though just slightly superior, these clustering results are achieved with considerably smaller representations (at least seven times smaller) than that used in [13,14]. We therefore conclude that talking points capture not just meaningful aspects of a lexical concept, but the most salient aspects of a concept.

6 Conclusions

Creative linguistic devices like metaphor are knowledge-hungry in the extreme, since

they exploit both a factual knowledge of the world and a knowledge of how these facts, or talking points, can be nudged into the realm of the colorful, the fanciful and the resonant. Any computational treatment of metaphor will thus only be as good as the knowledge representation that supports it. The representation described here – called *talking points* – is simple yet scalable, and lends computational substance to some key insights in the metaphor literature, from the interaction theory of Black [3] to the conceptual blending theory of [4] as computationally modeled by [7]. We also employ a key insight from the work of Hofstadter and his fluid analogies group [6], that creative reasoning on a conceptual level requires a degree of meaning slippage that must be supported by the underlying knowledge representation.

Our knowledge-base of talking points is derived from two complementary information sources: the objective definitions contained in WordNet [2] and the stereotypical comparisons that pepper the texts of the web [12]. These sources yield a knowledge-base that is neither small nor hand-crafted. While the knowledge-base needs to grow by at least an order of magnitude, slippage means that non-identical talking points can be treated as equivalent for purposes of robust processing, which in turn extends the *halo* of talking points that surrounds each concept in the knowledge-base [6]. Our replication of the experiments of [13,14] also indicates that, in a pinch, new talking points for a previously under-represented concept can be acquired dynamically from the web with reasonable accuracy. As it currently stands, the talking points approach to metaphor is robust and scalable enough to generate simple but imaginative metaphors on demand for a wide range of user inputs.

References

1. Indurkhya, B.: Metaphor and Cognition: Studies in Cognitive Systems. Kluwer. (1992)
2. Fellbaum, C. (ed.): WordNet: An electronic lexical database. MIT Press. (1998)
3. Black, M.: Models and Metaphor: studies in language and philosophy. Cornell Uni. (1962)
4. Fauconnier, G., Turner, M.: Conceptual Integration Networks. Cognitive Science, 22(2), 133–187. (1998)
5. Collins, A., Loftus, E. F.: A Spreading-Activation Theory of Semantic Processing. Psychological Review 82, 407-428. (1975)
6. Hofstadter, D.R., Fluid Analogy Research Group: Fluid Concepts and Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought. Basic Books (1994).
7. Veale, T., O'Donoghue, D.: Computation and Blending. Cog. Linguistics, 11(3-4). (2000)
8. Wilks, Y.: Making Preferences More Active. Artificial Intelligence 11(3), 197-223. (1978)
9. Fass, D.: Met*: a method for discriminating metonymy and metaphor by computer. Computational Linguistics, 17(1), 49-90. (1991).
10. Martin, J.: A Computational Model of Metaphor Interpretation. Academic Press. (1990)
11. Barnden, J. A., Lee, M. G.: An Artificial Intelligence Approach to Metaphor Understanding. Theoria et Historia Scientiarum, 6(1), 399-412. (2002)
12. Veale, T., Hao, Y.: Comprehending and Generating Apt Metaphors: A web-driven, case-based approach to figurative language. Proc. of the 22nd AAAI conf. on AI, 1471-76. (2007)
13. Almuhareb, A., Poesio, M Attribute-Based and Value-Based Clustering: An Evaluation. Proc. of EMNLP, Emerging Methods in NLP, 158--165. (2004)
14. Almuhareb, A., Poesio, M.: Concept Learning and Categorization from the Web. Proc. of the annual meeting of the Cognitive Science society. (2005)

Automatic Composition of Themed Mood Pieces

Heather Chan and Dan Ventura

Department of Computer Science
Brigham Young University
Provo, Utah, USA
heatherchan@byu.net, ventura@cs.byu.edu

Abstract. Musical harmonization of a given melody is a nontrivial problem; slight variations in instrumentation, voicing, texture, and bass rhythm can lead to significant differences in the mood of the resulting piece. This study explores the possibility of automatic musical composition by using machine learning and statistical natural language processing to tailor a piece to a particular mood using an existing melody.

Key words: Composition, Harmonization, Creativity, Mood

1 Introduction

There are many examples of thematic musical works in which the composer takes a single melody and reworks it in different ways to produce a variety of related material. This practice is particularly useful in soundtracks for programmatic works such as films, TV series, and video games. By referring back to and reusing the same melodic elements, the composer is able to tie together several otherwise unrelated pieces so that listeners understand and associate with a specific storyline and set of characters. Examples of such works include the Lord of the Rings movie trilogy, the Simpsons TV series, and the Zelda game series.

The reworking of an existing theme is a skill subset of composition that demands creativity from the composer, yet can be viewed as a simpler task than composition from scratch. We are interested in the ability of software to emulate this compositional practice in order to produce an extensive set of musical material for use in the aforementioned genres. In particular, we wish to examine the feasibility of tailoring individual pieces to specific moods based on user request.

Approaches to automatic harmonization include n-gram statistical learning for learning musical grammars [1], genetic algorithms for generating four-part harmony [2], and hidden Markov models for chorale harmonization [3]. We have chosen to concentrate on statistical methods, which can apply one of at least four basic methods of music generation: random walk, HMM with Viterbi decoding, stochastic sampling and pattern-based sampling [4]. Chuan and Chew and Whorley *et al.* have investigated the automatic generation of style-specific accompaniment for melodies [5, 6]. This is very similar to what we hope to accomplish in that their systems address melodies that have already been composed

and attempt to provide a well-written accompaniment for each of them. However, while their systems concentrate on musical style as the main determining factor of their harmonizations, we wish to focus on mood and emotional impact. Considering the motivational/methodological ontology suggested by Pearce *et al.* [7], we suggest that our proposed system be considered as an approach to algorithmic composition with perhaps some shading towards the design of a compositional tool.

We seek to design a system that takes an existing melody as input and produces as output an automatically composed piece of music based on user-specified parameters governing the overall mood or feel of the piece. We limit our scope to melodies in regular meters that do not contain non-chord tones on strong beats and that do not call for key modulation in the harmony.

Many systems for automatic musical composition have been met with criticism for being either too biased by the tastes of the creators or, conversely, too broad or arbitrary in their definitions of music to be able to meet the musical tastes of the audience. We aim to develop a system that demonstrates a higher level of creativity than these by creating independently and creating something of value.

2 System Design

Our design requires four functions: a *melody filter* to select the melody notes we will use to harmonize; a *chord progression generator* to produce a suitable chord progression for the melody subset; a *harmonization planner* to select the instrumentation and harmonization technique to be used based on the mood; and a *composer* to build the piece from the melody, the generated chord progression, and the chosen instrumentation and harmonization technique (see Fig. 1).

2.1 Melody Filter

For now, we implement the melody filter as simply as possible. We specify a sampling interval based on measure length over which a single chord will persist, based on strong beats, and we consider only the first melody note in each interval as significant in chord assignment. An example is shown in Figure 2.

2.2 Chord Progression Generator

The chord progression generator requires a more complex approach. To harmonize a given melody, one must consider not only the melody notes but also which chords are compatible with them and which of those chords fit together well. We can treat the melody notes as observed events and the underlying chord progression as a hidden state sequence, modeling the inference problem as a simple hidden Markov model (HMM) (see Figure 3). Here, we must determine the chord progression from the melody notes, similar to the process of Roman numeral analysis used by music theory students. The first-order HMM depicted

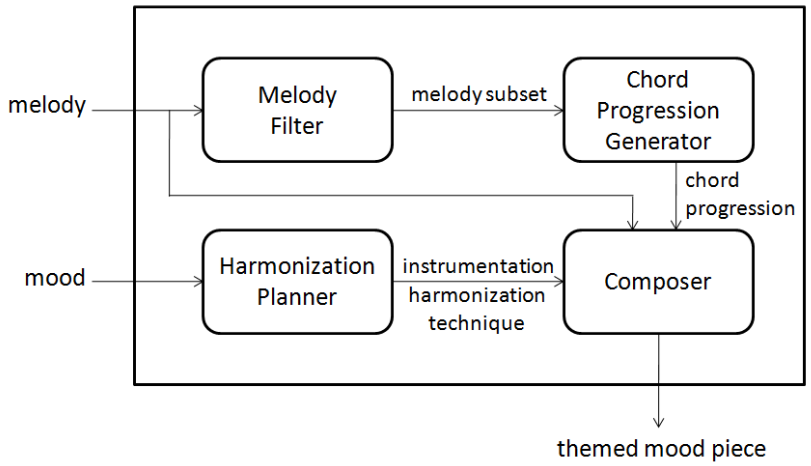


Fig. 1. *Block diagram of system design.* Given an input melody, the *melody filter* selects the melody notes that will be used to harmonize. Those notes are passed to the *chord progression generator* which produces a suitable chord progression for the melody subset. The *harmonization* planner selects the instrumentation and harmonization technique to be used based on the desired mood. Finally, the *composer* builds the piece from the melody, the generated chord progression, and the chosen instrumentation and harmonization technique.

in Figure 3 should be sufficient for modeling well-formed chord progressions, as classical music theory suggests that the extent to which each successive chord is pleasing to the ear is heavily conditioned on its relationship to the immediately preceding chord.

The parameters of the model represent two conditional probability distributions: the next chord given the preceding chord $P(c_i|c_{i-1})$ and the observed melody note given the current chord $P(m_i|c_i)$. We expect the former to demonstrate that the dominant almost always resolves to the tonic, and occasionally to the submediant; the latter should reflect the low probability of observing the second scale degree in the melody line during a tonic chord. These distributions can be modeled using statistics gathered from MIDI datasets.

Given the melody, we can use statistical inference to sample the distribution $P(c_i|m_i)$, giving us, for example, a maximum likelihood estimate of the generating chord progression; we have chosen the Viterbi algorithm for now to accomplish this[1]. The previous example is continued in Figure 4. When a suitable chord progression has been determined, we can build harmonizations using common accompanimental figures.

[1] We limit our study to diatonic chords with no key modulation.

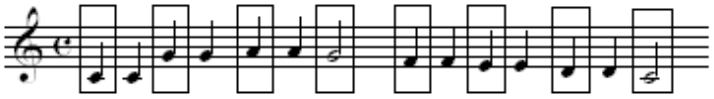


Fig. 2. *Melody filtering.* "Twinkle, Twinkle Little Star" is sampled in half-measure intervals and only the sampled notes are considered significant for determining chord progression.

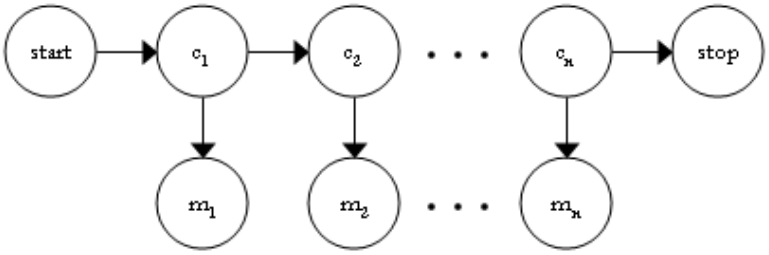


Fig. 3. *Graphical model used for chordal inference and progression generation.* The nodes labeled c_1, c_2, ..., c_n represent the hidden chords, and the nodes labeled m_1, m_2, ..., m_n represent the observed (sampled) melody notes.

2.3 Harmonization Planner

We view the implementation of the harmonization planner as a machine learning problem. Initially, we have limited our choice of instrumentation to piano accompaniment, string accompaniment, harp accompaniment, and electric guitar accompaniment. We would like to be able to specify a mood to the system such as "joyous", "urgent", "tragic", etc. and have the system compose a suitable piece for that mood. In order to learn the appropriate parameters, we need to rely on human feedback, as we have no way of determining the effective mood of a piece without referring to popular opinion.

We will investigate two different approaches to learning the relationship between instrumentation, harmonization, and mood. The first method will involve generating several pieces of music using different instrumentations and harmonizations as input and asking listeners to categorize the resulting pieces into different classes of moods; we can then cluster the examples and select a nearest neighbor example of instrumentation and harmonization when a user requests a specific mood. This will require a way to define a distance measure between

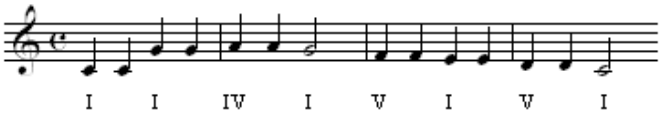


Fig. 4. *Chord progression.* Sampling the distribution $P(c_i|m_i)$ results in a possible generating chord sequence for "Twinkle, Twinkle Little Star".

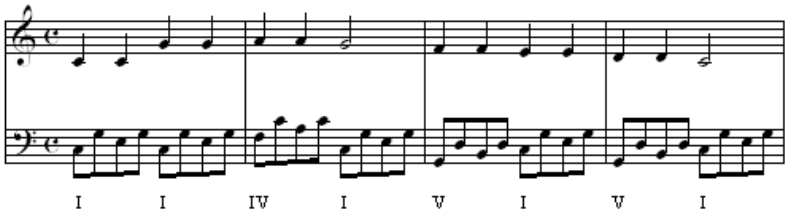


Fig. 5. *Composition.* An arpeggiated harmony for "Twinkle, Twinkle Little Star".

pieces that will yield useful clusters, possibly through a trial-and-error weighting of the inputs and manual inspection of the resulting clusters.

The second method will involve having the user specify a mood, generating several pieces of music as before, and having the user choose the n best pieces that come the closest to the mood effect that the user had in mind; the input would be the specified mood, and the outputs would be the specific instrumentation and harmonization technique used. This is a single-input multiple-output learning problem that may be difficult to learn but may be amenable to solution by neural networks or by application of inverse methods.

2.4 Composer

Once the chord progression, instrumentation, and harmonization technique have been determined, the final piece can be written. This is a simple matter of setting the voices according to the selected instrumentation, expanding the chords in the chord progression using the chosen harmonization technique, and combining the resulting accompaniment with the original melody. The concluding result of our example is shown in Figure 5.

3 Assessment of Creativity

The artist's ambition is not to discover a "correct" answer, but a "creative" one. We have therefore decided to approach the evaluation of this system not as a measure of accuracy, but as one of creativity.

Recently it has been suggested that for a computational system to be considered creative, it must be perceived as possessing *skill*, *appreciation*, and *imagination* [8]. We can describe the system as skillful if it exhibits knowledge of musical behavior and can make intelligent decisions in composition that lead to a well-formed piece of music. We have provided our system with statistical information about accepted progressions and behavior via an HMM, and we are implementing compositional techniques for harmonization including a few different forms of chordal accompaniments. We thus argue that this system is skillful.

We can describe the system as appreciative if it consistently produces something of value and can evaluate itself to some extent in order to adapt its work to the user's tastes. We are currently working on using machine learning algorithms and user feedback to discover associations between various instrumentations and harmonization techniques and the moods they tend to produce. We hope to amass enough data from user feedback to make this possible; if we succeed, we will be able to argue that this system is appreciative.

We can describe the system as imaginative if it creates new material that demonstrates some level of independence both from its creator's designs and from works by other composers. We have introduced a bias towards particular datasets by relying on statistical methods to ensure that the system produces plausible chord progressions; we have also introduced a bias towards the creator's compositional style because the system draws only on instrumentations and harmonization techniques that are programmed. However, these influences are acceptable to some degree since all composers are influenced both by their mentors and by other composers whose works they have studied. In addition, this system has two creative abilities that help offset bias: it is able to generate chord progressions that have never been seen before, and it selects combinations of instrumentations and harmonization techniques based on knowledge of user feedback, which is independent both of the creator's tastes and of the tastes of other composers. We therefore describe this system as somewhat imaginative.

Although it is difficult to quantify creativity, we can survey audiences and obtain numerical ratings for each of these characteristics. Questions might include: "On a scale of 1 to 10, how much does this sound like real music?" "On a scale of 1 to 10, how closely does this match the mood you were thinking of?" "On a scale of 1 to 10, how much does this sound like something you have heard before?" (skill, appreciation, and imagination, respectively). In addition to being informative on their own, these ratings might be useful data for training the system to self-evaluate its creativity and increase its level of appreciation.

Our system is skillful and somewhat imaginative, but still lacks appreciation for the value of what it can create. We hope to remedy this by exposing the system to more user feedback in the future in order to help it learn what is of value and what is not.

4 Discussion

We have proposed a system for automatic composition of mood pieces based on an existing melody and have presented our approach to constructing such a system. Currently, the system design is limited to diatonic melodies that do not modulate in key, but we plan to accommodate greater harmonic complexity and key modulation in the input melodies in the future, expanding the system's harmonic vocabulary to include borrowed and altered chords and investigating the use of a key-finding algorithm to detect key modulation.

Of course, greater harmonic complexity will significantly increase the size of the joint space, making our inference task computationally challenging. To address this issue, we plan to incorporate stochastic inference methods to replace the use of the Viterbi algorithm. As an additional benefit, the stochastic element will allow for nondeterminism in the selection of an appropriate chord progression, enhancing the system's creative ability.

We are currently limited in our choice of instrumentations and harmonization techniques, and we would like to incorporate percussion rhythms and effects, as these are often crucial in heightening the emotional intensity of a piece. If possible, we would also like to implement other tools of variation such as change of register, change of meter, change of mode, and melody augmentation/diminution.

Finally, as it is difficult to gather data for the mood-learning portion of this problem (since it must be obtained from a human audience), we require an efficient method of obtaining significant amounts of data on the correlations between instrumentation, harmonization, and mood.

References

1. Ponsford, D., Wiggins, G., Mellish, C.: Statistical learning of harmonic movement. Journal of New Music Research **28**(2) (1998) 150–177
2. Phon-Amnuaisuk, Wiggins, G.: The four-part harmonisation problem: a comparison between genetic algorithms and a rule-based system. In: Proceedings of the AISB99 Symposium on Musical Creativity. (1999)
3. Allan, M., Williams, C.K.I.: Harmonising chorales by probabilistic inference. In: Advances in Neural Information Processing Systems 17. (2005) 25–32
4. Conklin, D.: Music generation from statistical models. In: Proc. AISB Symposium on Artificial Intelligence and Creativity in the Arts and Sciences. (2003) 30–35
5. Chuan, C., Chew, E.: A hybrid system for automatic generation of style-specific accompaniment. In: Proc. Int. Joint Workshop on Computational Creativity. (2007) 57–64
6. Whorley, R.P., Wiggins, G.A., Pearce, M.T.: Systematic evaluation and improvement of statistical models of harmony. In: Proceedings of the International Joint Workshop on Computational Creativity. (2007) 81–88
7. Pearce, M., Meredith, D., Wiggins, G.: Motivations and methodologies for automation of the compositional process. Musicae Scientiae **6**(2) (2002) 119–147
8. Colton, S.: Creativity vs. the perception of creativity in computational systems. In: Creative Intelligent Systems: Papers from the AAAI Spring Symposium, Technical Report SS-08-03, AAAI Press (2008) 14–20

A Computer Model for the Generation of Monophonic Musical Melodies

Juan Alvarado López[1] and Rafael Pérez y Pérez[2]

[1] Posgrado en Ciencias e Ingeniería de la Computación
Universidad Nacional Autónoma de México
jalvarado@uxmcc2.iimas.unam.mx
[2] Departamento de Tecnologías de la Información
Universidad Autónoma Metropolitana, Unidad Cuajimalpa
México D.F.
rpyp@rafaelperezyperez.com

Abstract. This paper describes a system that generates monophonic melodies called ERMEG (Engagement and Reflection MElodies Generator). It is based on the engagement-reflection computer model of creativity. ERMEG performs two main processes: the creation of knowledge structures and the generation of a new melody (E-R cycle). During engagement the system produces sequences of musical phrases driven by content and musical-theory constraints and avoids the use of explicit goals or predefined melody-structures. During reflection the system breaks impasses, verifies the coherence of the melody in progress and evaluates the novelty of the material produced so far and, as a result of this evaluation, it generates a set of guidelines that work as constraints during engagement.

Key words: Creativity, Music, Automatic Composition, Engagement, Reflection.

1 Introduction

This paper describes a computer program named ERMEG (Engagement and Reflection MElodies Generator) which develops monophonic melodies. It is based on the Engagement-Reflection computer model of creativity (E-R model) which has been employed to develop a computer program for plot generation known as MEXICA [1]. The main characteristics of the E-R model are:

- The user provides a set of examples that are employed to build the system's knowledge structures.
- The model includes two core processes: a generation process (known as engagement) and an evaluation processes (known as reflection). During engagement the system generates material guided by content constraints; during reflection the system evaluates the material generated so far -modifies it if it is necessary- and as a result of the evaluation adjusts the constraints that drive the generation of material during engagement. The system's outputs are the result of the interaction between engagement and reflection.

- An important characteristic of the model is that, following Gelernter [2], emotions are the glue that joins ideas during engagement. Thus, in MEXICA, emotional links and tensions between characters drive the generation of material during the unraveling of the plot [3].
- The model provides mechanisms to control the behavior of the different processes that conform the E-R cycle.

The main purpose of this work is to explore if the ideas that inform the E-R model employed for building MEXICA can be applied for musical composition of monophonic melodies. We believe that the study of the commonalities and differences between plot generation and melodies generation employing the E-R model might result in interesting information about computational creativity. In this paper we focus in describing how we implemented the E-R model for producing monophonic melodies. Our work considers that:

- A melody can be divided in sequences of notes; we refer to them as composition-phrases.
- Composition-phrases can be associated with representations of emotions and tensions.
- These representations of tensions and emotions can be employed during engagement as cue to probe memory in order to generate a sequence of musical phrases that conform a monophonic melody.
- Following the original E-R model, the user provides a set of examples to build its knowledge structures.

In this document we describe how the system builds its knowledge structures, how we associate emotion and tensions to musical phrases and how the engagement-reflection cycle works. We show some examples of melodies produced by the system and provide some discussion about the system.

2 Creation of Knowledge Structures.

ERMEG builds its knowledge structures from a set of melodies provided by the user of the system called Previous Melodies. Each Previous Melody is divided by the system into groups of notes that we refer to as composition-phrase (CP). The length of each CP is defined by the user of the system in terms of beats. That is, the user might ask the system to create one CP every four beats. So, the notes included in the first four beats of the first Previous Melody constitutes the CP1; the notes included in the second four beats of the first Previous Melody constitutes the CP2; and so on. The process is repeated for each Previous Melody. As we will explain later, Composition-Phrases are an essential element during engagement. Because the user has the possibility of modifying CP's length, it is possible to compare ERMEG's outputs for different lengths.

Each CP has a set of three characteristics we are interested about:

- Speed: Every phrase has associated a Speed which can have any of the following values: Fast, Medium and Slow. These values depend on the number of notes in the phrase and its length (in beats). For example, if we have eight notes in two beats the phrase is classified as Fast; on the other hand, if we have three notes in eight beats, the phrase is classified as Slow. The system employs a predefined table to determine these relations.
- Interval-Difference is defined as the difference (in intervals) between the first and last notes in a phrase. It can have the following values: Big, Medium, Small.
- Interval-Variation: The Interval-Variation is defined as the average distance (in intervals) between all the notes within a phrase. Its possible values are Big, Medium and Small.

We employ these three characteristics to associate to each CP a value of tension and emotion.

2.1 Tension and Emotion

In the literature one finds different studies about music and emotion. For example, Dalla Bella [4] points out how the tempo strongly influences the type of emotions triggered by a melody: fast tempos tend to evoke happiness and slow tempos tend to evoke sadness. Khalfa et al. [5] have performed experiments that illustrate that neither the tempo nor the rhythm alone generate strong emotions; it is the combination of both, rhythm and melody, that trigger emotions in people. Juslin and Sloboda [6] have shown that tonal music in high pitches and fast tempo can generate happiness; that big variations (leaps) of pitch and high pitches can generate excitation or anxiety; that low pitches and slow tempo can generate sadness.

Based on these works, we decided to employ Speed, Interval-Difference and Interval-Variation to associate emotional and tensional characteristics to our musical phrases. The current version of the system employs one type of emotion -which represents a continuum between happiness and sadness- and one type of tension -which represents a continuum between anxiety and calmness. (We are aware that emotions in music are more complex than this. But for our first prototype we have decided to use a simple representation of emotions). The system calculates the tension and emotion by means of fuzzy logic. The values of Speed, Interval-Difference and Interval-Variation are the input to the fuzzy system. The following lines explain how these values are obtained. Suppose that we have an eight notes phrase with a length of two beats (see Figure 1).

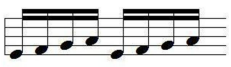


Fig. 1. An example of a phrase.

The Speed is calculated dividing the number of notes in the phrase by its length in beats:

$$Speed = \frac{Notes}{Length} \Rightarrow Speed = \frac{8}{2} \Rightarrow Speed = 2$$

The Interval-Difference, the difference (in intervals) between the first and last notes, is equal to $2\frac{1}{2}$ tones (the first note in the phrase is a E and the last note is A):

$$IntervalDifference = 2.5$$

As explained earlier, the Interval-Variation is the average distance (in intervals) between all the notes within a phrase. In this case, the distance between the first and the second notes is equal to $\frac{1}{2}$ tone, the distance between the second and the third notes is equal to 1 tone, the distance between the third and the fourth notes is equal to 1 tone, and so on:

$$IntervalVariation = \frac{.5 + 1 + 1 + 2.5 + .5 + 1 + 1}{7} = 1.071...$$

Now, we can apply the fuzzy logic (see figure 2). We obtain the following membership values (MV):

Speed Fast with MV = 1.0	Variation Small with MV = 0.428
Difference Small with MV = 0.583	Variation Medium with MV = 0.571
Difference Medium with MV = 0.416	

The minimum and maximum values of the universe of discourse of the variables Speed, Variation and Difference are determined experimentally.

The next step is to apply a set of fuzzy rules (we have defined 54 rules); the implication operator we employ is Mamdani and the aggregation operator is union. For this example, the rules selected for emotion are:

1. IF Speed IS Fast AND Difference IS Small AND Variation IS Small THEN Emotion = Happy
2. IF Speed IS Fast AND Difference IS Small AND Variation IS Medium THEN Emotion = Happy
3. IF Speed IS Fast AND Difference IS Medium AND Variation IS Small THEN Emotion = Happy
4. IF Speed IS Fast AND Difference IS Medium AND Variation IS Medium THEN Emotion = Very Happy

And the rules selected for tension are:

1. IF Speed IS Fast AND Difference IS Small AND Variation IS Small THEN Tension IS Very Calm
2. IF Speed IS Fast AND Difference IS Small AND Variation IS Medium THEN Tension IS Calm
3. IF Speed IS Fast AND Difference IS Medium AND Variation IS Small THEN Tension IS Neutral
4. IF Speed IS Fast AND Difference IS Medium AND Variation IS Medium THEN Tension IS Neutral

Employing the membership values for Speed, Difference, and Variation, we obtain the following results:

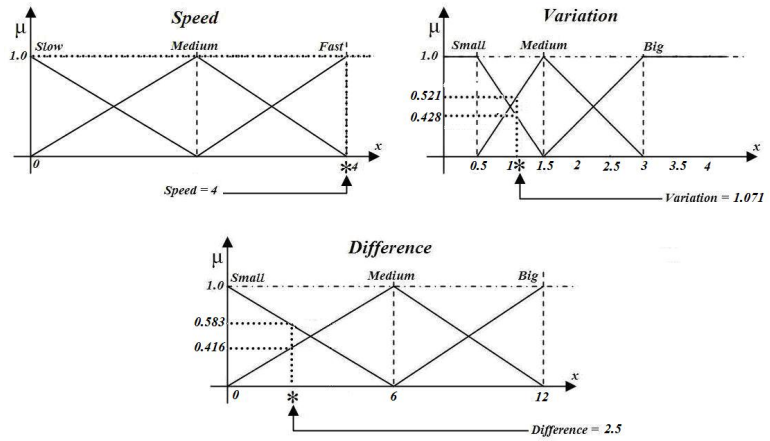


Fig. 2. The values of Speed, Difference and Variation, and their membership values for the fuzzy sets.

Emotion	Tension
Rule1 = Happy with MV = 0.428	Rule1 = Very Calm with MV = 0.428
Rule2 = Happy with MV = 0.571	Rule2 = Calm with MV = 0.571
Rule3 = Happy with MV = 0.416	Rule3 = Neutral with MV = 0.416
Rule4 = Very Happy with MV = 0.416	Rule4 = Neutral with MV = 0.416

The next step is to perform the defuzzification using the Center Of Area (COA) method. So, having in mind the values of the singletons (Happy=80, Very_Happy=100, Very_Calm=20, Calm=40 and Neutral=60) the calculation is performed as follows:

$$Emotion = \frac{(Happy*0.428)+(Happy*0.571)+(Happy*0.416)+(Very\_Happy*0.416)}{0.428+0.571+0.416+0.416}$$

$$Emotion = 84.54$$

And for the tension:

$$Tension = \frac{(Very\_Calm*0.428)+(Calm*0.571)+(Neutral*0.416)+(Neutral*0.416)}{0.428+0.571+0.416+0.416}$$

$$Tension = 44.41$$

Finally, the value of the tension is modified by the average octave of the phrase. This occurs because the higher the octave is the higher the tension that the phrase produces. So, for this example the average octave of the phrase is 4, therefore the tension is equal to 177.64:

2.2 Building the Knowledge Base

In ERMEG a Previous Melody is defined as a sequence of CPs:

$$\text{Previous-Melody} = \text{CP1} + \text{CP2} + \text{CP3} + \ldots \text{CPn}$$

Each CP has associated emotional and tensional values, which we refer to as Local-Emotion and Local-Tension. But from CP2 onwards we can also associated to each phrase a Historical-Emotion and a Historical-Tension. The Historical-Emotion is a vector that records the value of the Local-Emotion of each phrase that has preceded the current CP; the Historical-Tension is a vector that records the value of the Local-Tension of each phrase that has preceded the current CP. In this way, the Historical-Emotion of CP3 is equal to [Local-Emotion of CP1, Local-Emotion of CP2].

All these information is recorded into a structure known as Context; so, the Context is comprised by four elements: the local emotional and tensional values of the current CP, and its historical emotional and tensional vectors. Each time a CP is performed, its emotional and tensional values are calculated and the Context is updated. We refer to the content of the Context after CP1 is performed as Context1; we refer to the content of the Context after CP2 is performed as Context2; and so on (see Figure 3).

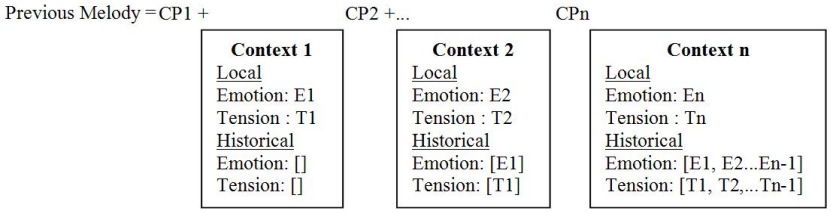


Fig. 3. Representation of a Previous Melody and its Contexts.

Then, the system links contexts with the next CP in the melody. So, Context1 is associated with CP2, Context2 is associated with CP3, Context(n-1) is associated with CPn. The purpose of this type of associations is to establish possible ways to continue a composition given a specific context. In this way, the system records information like "when a melody has a context like Context 1, a possible way to progress the melody is performing CP2". With this information, the system creates in memory its knowledge structures -known as atoms- that it will employ during the engagement-reflection cycle to generate novel melodies.

Thus, in order to create its knowledge structures, the system performs the following process for each Previous Melody.

1. The system initializes the Context and makes the first phrase in the melody the current CP.

2. The system calculates its Emotional and Tensional values and updates the Context.
3. The content of the Context is copied into a new structure known as Atom.
4. The system associates the following CP in the melody to the Atom; we refer to this phrase as the next possible action to be performed in the melody.
5. The system takes the next CP in the melody, makes it the current CP and goes back to step 2 until all phrases in the melody are processed.

3 The Engagement-Reflection Cycle.

ERMEG generates melodies as a result of an engagement-reflection cycle.

3.1 Engagement

The engagement cycle works as follows:

1. The user provides an initial phrase.
2. The system calculates its emotional and tensional values and updates the Context. This is the initial Context.
3. The Context is employed as cue to probe memory. The system looks for all atoms which are equal or similar to the current Context and retrieves their associated CP.
4. The system selects at random one of the CPs retrieved as the next (musical) action to be performed in the melody in progress.
5. The selected CP is appended to the melody in progress, the Context is updated, and the cycle starts again (it goes to step 3).

Engagement ends when an impasse is declared (i.e. no atom can be matched), or when a predefined number of CPs have been added to the melody in progress.

During engagement the system employs two important parameters to decide if a Context matches an Atom (i.e. to decide if the Context is equal or similar to the Atom): the Resolution-Constant and the ACAS-Constant. The Resolution-Constant determines if two values are considered as equivalent. For example, if the Resolution-Constant is set to 90%, the Local-Emotion in the Context is equal to 95 and the Local-Emotion in an Atom is equal to 100, they are considered equivalents. That is, any value between 90 and 100 is considered as equivalent to 100. The ACAS-Constant indicates the minimum percentage that the Context must be equivalent to the Atom in order to produce a match. For example, if the ACAS-Constant is set to 50%, at least the 50% of the Context must be equivalent to the Atom. The values of these constants have important effects during engagement. If the Resolution-Constant is set to 100% and the ACAS-Constant is set to 100%, we are forcing the system to reproduce any of the Previous Melodies. On the other hand, if the Resolution-Constant is set to 1% and the ACAS-Constant is set to 1%, the system is forced to select at random between all the CPs in its knowledge-base the next phrase to continue the melody.

3.2 Reflection

During Reflection the system tries to break impasses and evaluates the coherence and novelty of the melody in progress.

Breaking Impasses. When an impasse is declared, the system decrements in 10% the value of the Resolution-Constant and the ACAS constant, and then switches back to engagement. The purpose is to make easier to match an atom. If after modifying the constants the system cannot match any atom, the system tries a second strategy: it selects at random a CP from its knowledge base, added it to the melody in progress and switches back to engagement. The purpose of this strategy is to modify the Context in a way that now it can match an atom. If the system fails again the impasse is declared as unbreakable and the E-R cycle ends.

Evaluation of Coherence. In ERMEG a melody is coherent when there are not abrupt jumps of notes between two continuous phrases within a melody. For example, let us imagine that the last note of CP1 is the first degree and the first note of CP2 is the seventh degree; in this case, the distance between them is too big ($5\frac{1}{2}$ tones) and therefore the coherence is broken (the maximum allowable distance between two continuous phrases can be modified by the user of the system). To solve this problem the system moves the second phrase closer to the first one, or creates a new CP that works as a bridge that joins both phrases in a smooth way. For example, suppose that engagement generates the two phrases in Figure 4. Reflection evaluates that composition as incoherent. The system modifies the melody either by moving back the second phrase closer to the first one and in this way reducing the distance between them (Figure 5.a), or by building a bridge of notes in two measures that will join the two phrases (Figure 5.b).

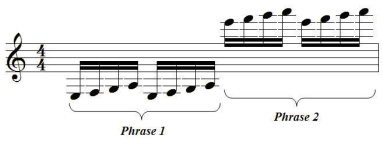


Fig. 4. Two phrases generated by engagement.

Evaluation of Novelty. In this work, a melody is considered original when it is not similar to any of the Previous Melodies. The system compares the melody in progress with all Previous Melodies. If they are similar the system decrements the value of the Resolution and ACAS Constants in order to look for novel atoms to match.

The engagement-reflection cycle ends when an impasse is unbreakable, or when the system performs a predefined number of E-R cycles.

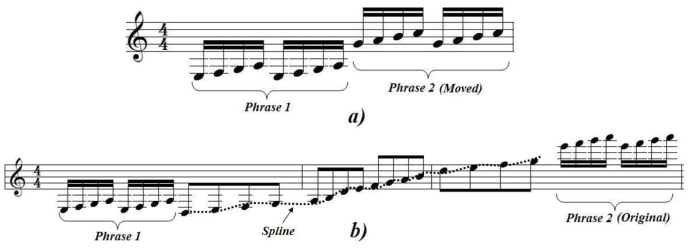


Fig. 5. Modification made by reflection.

4 Discussion.

Figures 6 and 7 show two melodies generated by ERMEG. The system performed the same number of engagement-reflection cycles for each of the examples. The following parameters were employed in both cases:

- Resolution-Constant : 90%
- ACAS-Constant: 50%
- Maximum difference allowed between two consecutive notes in two different phrases: 3 Tones

ERMEG employed the same group of Previous Melodies to generate the two melodies. However, for the melody in Figure 6, the system employed CPs with a length of 8 beats and for the melody in Figure 7 the system employed CPs with a length of 2 beats. In both figures, the first measure corresponds to the initial phrase provided by the user, which was the same for both examples; the shaded parts in the figures show those notes inserted or modified during reflection; the rest are the phrases generated during engagement.

Although the melodies shared the same initial phrase, and the same group of Previous Melodies, they progressed in different ways. There were two main reasons for this behavior: the content of Atoms depended on the size of the CPs; the number of Atoms, and therefore the number of options to progress a melody, also depended on the size of the CPs. Thus, each time we modified the length of the CPs, we altered the content and number of knowledge structures inside the system. In the case of the melody in Figure 6 it was easy to recognise the origin (in the set of Previous Melodies) of each phrase. This made sense because the building blocks, i.e. the CPs, were quite long: 8 beats. However, in the case of the melody generated with CPs equal to 2 beats (Figure 7), it was hard to identify where each phrase came from.

The main purpose of this work was to explore if the ideas that inform the E-R model employed for building MEXICA could be applied for musical composition. ERMEG suggests that the E-R model can be employed to generate novel monophonic melodies.

Music composition is a very complex process. It involves several aspects that have not been taken into consideration in this work. There are several computer

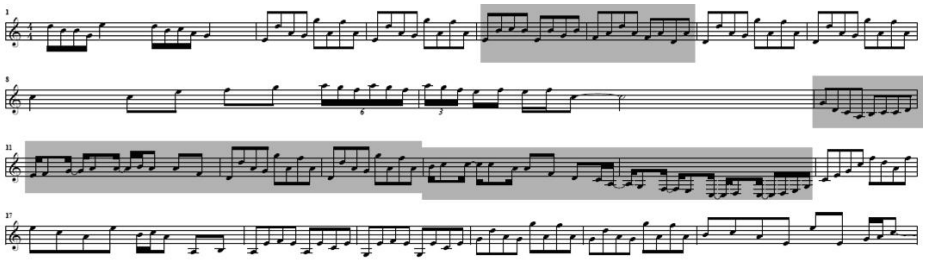


Fig. 6. A melody generated by ERMEG with a composition-phrase equal to 8 beats.

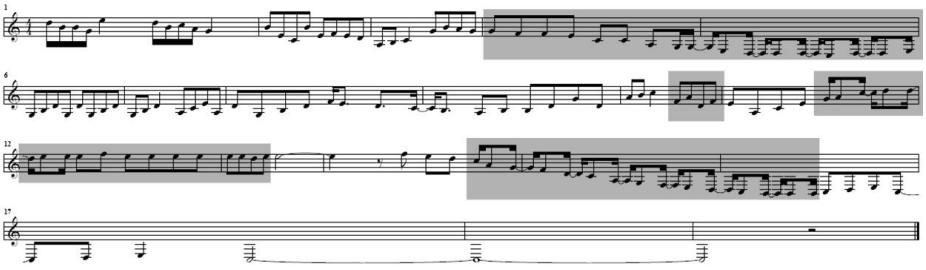


Fig. 7. A melody generated by ERMEG with a composition-phrase equal to 2 beats.

programs that produce very interesting results in this area. We are interested in generating a computer model of creativity that can be applied in different domains. We believe this work is a good first step towards a more complete E-R Model.

References

1. Pérez y Pérez, R & Sharples, M. (2001). MEXICA: A computer model of a cognitive account of creative writing. *Journal of Experimental and Theoretical Artificial Intelligence*, 13, 119-139.
2. Gelernter, D. 1994. *The Muse in the Machine*. London: Fourth Estate.
3. Pérez y Pérez, R. (2007). Employing Emotions to Drive Plot Generation in a Computer-Based Storyteller. *Cognitive Systems Research*. Vol. 8, number 2, pp. 89-109.
4. Dalla Bella, S., Peretz, I., Rousseau, L., & Gosselin, N. (2001). A developmental study of the affective value of tempo and mode in music. *Cognition*, 80(3), B1-B10.
5. Khalfa, S., Roy, M., Rainville, P., Dalla Bella, S. & Peretz, I. (2008) Role of tempo entrainment in psychophysiological differentiation of happy and sad music? *International Journal of Psychophysiology*, vol. 68, pp. 17-26
6. Juslin, P. & Sloboda, J. A. (2001) *Music and Emotion: Theory and Research*. Oxford: Oxford University Press.

Experiments in Constraint-Based Automated Scene Generation

Simon Colton

Department of Computing, Imperial College, London, UK
sgc@doc.ic.ac.uk

Abstract. We investigate the question of automatic scene construction for visual arts applications. We have implemented a system which can automatically infer a user's intentions from a partially formed scene, express the inferences as constraints, and then use these to complete the scene. We provide some initial experimental results with the system in order to compare two approaches to constraint-based scene construction. This leads on to a discussion about handing over increasingly meta-level responsibility when building computationally creative systems.

Keywords: scene generation, constraint solving, visual arts

1 Introduction

We have for some time been building an automated painter called The Painting Fool (www.thepaintingfool.com), which we hope will one day be accepted as a creative artist in its own right. We initially concentrated on automating various non-photorealistic rendering processes including segmenting and abstracting images, and the simulation of the look and usage of various natural media such as pencils, paints and pastels. Recently, however, we have investigated more cognitive, higher level issues related to painting. Firstly, we set up an expert system which can choose a painting style appropriate to an emotional keyword describing an image of a person's face. For instance, given the keyword 'sadness', The Painting Fool would choose to use a palette of muted colours and simulate pastels and pencils, in an attempt to heighten the melancholy expressed in the final painting. The painting style chosen in response to the keyword 'happiness' is very different. Moreover, we combined The Painting Fool with a vision system able to detect the emotion of a sitter for a portrait [5].

We have also addressed the question of getting The Painting Fool to invent its own scenes for painting. We have so far concentrated on constructing scenes with multiple similar objects arranged in some intelligent fashion. Paintings of such scenes are very common, for instance fields of flowers were very popular subject matter to the Impressionists. In [3], we implemented an evolutionary approach to the construction of such scenes, and we further used the HR system [2] so that the combined system both invented a fitness function and evolved a scene which maximised the fitness, with some surprising results. While this approach was successful, it had the drawback that to describe a new type of scene required defining a weighted sum of correlations over the elements in the scenes. In addition, certain constraints on the scene could not easily be guaranteed.

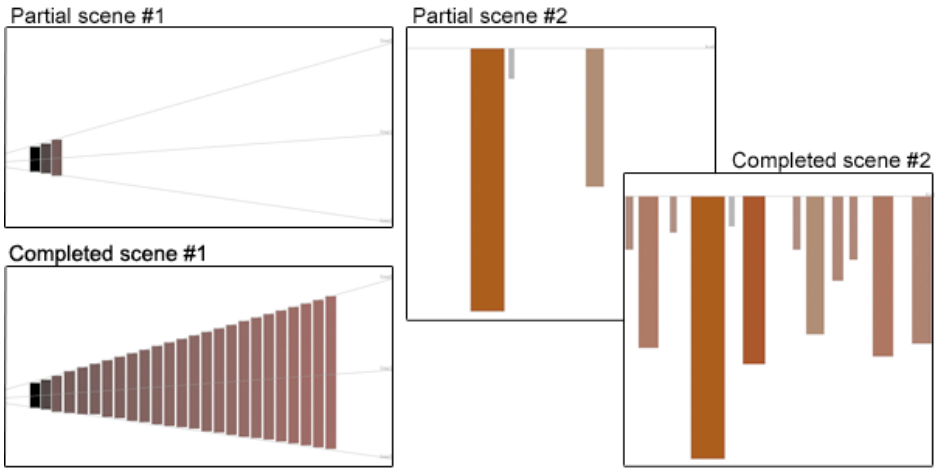


Fig. 1. Inspiring Scenes. In partial scene 1, three disjoint rectangles are arranged so that the tops, centres and bottoms are collinear, their widths and hues are equal, but both their brightness and saturation increase from left to right. In partial scene 2, three disjoint rectangles are arranged so that their tops are collinear and the rectangles with bases nearer the top of the scene are slimmer, shorter and less saturated than those with bases nearer the bottom of the scene.

We look here at an alternative approach to scene construction which attempts to address these drawbacks. In section 2 we describe how our system is able to induce a constraint satisfaction problem (CSP) from a partial scene supplied by a user. We also describe two ways in which the CSP can be used to generate scenes that contain elements adhering to the constraints exhibited in the partial scene. In section 3, we describe some initial experimental results from using the system. The work presented here has some overlap with the automatic derivation of designs from sketches [8]; diagrammatic reasoning [7]; geometric modelling via constraint solving [6] and scene understanding in machine vision applications [10]. However, our main inspiration has been Cohen's AARON program [9]. Much of the perceived creativity of AARON comes from the fact that it paints imagined scenes. In section 4, we describe some differences between the system we are building and AARON, and we use this as a springboard to discuss the requirement of handing over increasingly meta-level responsibility when building creative systems.

2 Constraint Based Scene Generation

We restrict ourselves to scenes which contain just rectangles of different locations, shapes, sizes and colours such as those in figure 1. These constructions are minimal, and can be considered placeholders for elements of a scene which will be painted. For instance, the rectangles in completed scene 2 of figure 1 could

be replaced by images of tree trunks and painted to produce a woodland scene similar to that in Klimt's Beechwood Forest painting of 1903. It is certainly possible to evolve such scenes using the methods described in [3], so there is a degree of overlap with our previous evolutionary approach. However, unless the evolved scene completely maximised the fitness function, which is unlikely, there would be no guarantee that certain constraints required by the user would be upheld in the completed scene. For instance, having one rectangle fully or partially occlude another in scene 2 might ruin the aesthetic of the woodland scene. As described in section 5, we see the constraint approach and the evolutionary approach as complementing each other, and we envisage an overall system which is able to use these and other methods to generate sub-scenes in much larger, more complex scenes.

In addition to the guarantee of certain constraints holding, in order to rapidly train The Painting Fool with numerous scenes, another requirement we specified was that the user would train the scene generator visually. In practice, this means that the user moves and changes rectangles on screen until the constraints he/she wishes to be upheld are present in their partial scene. Then, the system infers various constraints and asks the user to discard any which are present but not desired (and allows the user to pre-empt this by turning off certain constraint deriving processes). In subsection 2.1, we describe how the constraints are derived and represented in the syntax of the CLPFD constraint solver [1], and in subsection 2.2, we describe how we introduce randomness to the scene description. The user dictates how many rectangles are needed to complete the scene, and to achieve this, the system uses the derived constraints to generate and solve either a single constraint satisfaction problem (CSP), or multiple CSPs, as described in subsection 2.3. In either case, the variables in the CSP relate to the x and y coordinates of each rectangle, along with their *height*, *width*, *hue*, *saturation* and *brightness*. The constraints relate the variables either of one rectangle (a *unary* constraint) or of two rectangles (a *binary* constraint). The ability to derive a specification for a scene further advances the work presented in [3], where the user had to explicitly specify a fitness function, or relied on the HR system to invent one. We plan to implement the ability to interpret the observed constraints as a fitness function, so that the evolutionary approach can also be driven by user supplied examples.

2.1 Deriving Constraints

We originally intended to use a descriptive machine learning system such as HR [2] to form a theory about the partial scene, and then mine the theory for potential constraints. However, we found that the kinds of constraints that need to be expressed are relatively simple, so theory formation was not needed. Instead, we wrote methods able to derive (a) constraints on the domains of variables (b) constraints expressing collinearity properties (c) constraints expressing general relationships between variables defining single rectangles and (d) constraints expressing relational properties between pairs of rectangles, as described below.

Domain restrictions

Any properties true of all the rectangles in a partial scene can be used to narrow down the domain of the variables. For instance, in partial scene 1 of figure 1, all the rectangles have the same width and hue. This can be expressed in a CLPFD constraint program by simply substituting the variable with the constant value. In addition, the system allows the user to constrain the values of variables to be within the bounds of those present in the partial scene. For instance, the heights of the rectangles in completed scene 2 of figure 1 are between the heights of the largest and smallest rectangles of partial scene 2, as the user specified this.

Collinearity constraints

The system looks at every triple of rectangles and checks for rough (i.e., within a radius of 20 pixels) collinearity of their centre points and eight points around their edges. For any line covering the same points on a triple of rectangles, it checks whether *all* the rectangles in the partial scene are similarly roughly collinear and if so, expresses this as a constraint. Any lines which are sufficiently similar to another are discarded, in order to keep the number of collinearity constraints down. In our early experiments, we found that having points fit exactly on lines often over-constrained the CSP. Hence, we relaxed the derived constraint to state that the y-coordinate is within 150 pixels of the line. Note that the scenes are of size 1000 by 1000 pixels, and – in line with standard graphics practice – the y-coordinate in scenes *increases* from top to bottom. As an example, in partial scene 1 of figure 1, the top left point of the three rectangles is $(66, 391), (94, 383)$ and $(122, 373)$ respectively from left to right. The system calculates the line between the first two points using the standard equation $(y-y_1) = \frac{(y_2-y_1)}{(x_2-x_1)}(x-x_1)$. In order to keep the variables of the constraints whole numbers, the equation is multiplied throughout by (x_2-x_1). In our example, the line is therefore expressed as $28(y-391) = (-8)(x-66) \equiv 8x+28y-11476 = 0$. The system then checks whether the third point is within a distance of 20 pixels from that line. As this is true in our example, the system generates a constraint which dictates that the y-coordinate of the top left hand corner of each rectangle in the scene should be within 150 pixels of this line. The unary constraint in this case is expressed in CLPFD syntax as follows:

```
unary_constraint(line0,S) :-
    get_x(S,X), get_y(S,Y),
    (X*8) + (Y*28) - 11476 #> -150, (X*8) + (Y*28) - 11476 #< 150.
```

Note that `S` represents the rectangle to be constrained and `get_x` and `get_y` are able to to determine the x and y coordinate variables of the rectangle. There are similar predicates available for the height, width and colour variables.

Intra-rectangle relational constraints

At present, we have only one intra-rectangle relational constraint, which relates the height and width of the rectangles, so that the rectangles produced as solutions to the CSP have similar shapes. For example, in partial scene 2 of figure 1, the height to width ratio of the rectangles is between 1:5.16 and 1:7.71. To produce the completed scene, the user chose the option of constraining the pro-

portions of all rectangles in the scene to be within these ratios. This constraint was expressed in CLPFD syntax as:

```
unary_constraint(height_width_ratio_min_max_516_771,S) :-
     get_height(S,H), get_width(S,W), H100 #= H * 100,
     (W * 516) #< H100, (W * 771) #> H100.
```

Inter-rectangle relational constraints

The system is also able to notice global relationships between the variables of pairs of rectangles. To do this, it first calculates various properties of each rectangle which supplement their defining properties. For each property P, it then calculates the sets: $R_{(P,=)} = \{(r_1, r_2) : P(r_1) = P(r_2)\}$, $R_{(P,<)} = \{(r_1, r_2) : P(r_1) < P(r_2)\}$ and $R_{(P,>)} = \{(r_1, r_2) : P(r_1) > P(r_2)\}$. Then, each instance of a subset relationship between these sets gives rise to a constraint. For instance, in partial scene 2 of figure 1, the following subset relationship was determined by the system: $R_{(bottommost-y,<)} \subseteq R_{(width,<)}$. That is, the system induced the hypothesis that if rectangle r_1 has a base that is higher in the scene than rectangle r_2, then r_1 also has less width than r_2. Remembering again that y-coordinates increase from top to bottom, this binary constraint is expressed as follows:

```
binary_constraint(bottom_above_implies_less_width,S1,S2) :-
    get_y(S1,Y1), get_y(S2,Y2), get_height(S1,H1), get_height(S2,H2),
    Bottom1 #= Y1 + H1, Bottom2 #= Y2 + H2,
    get_width(S1,W1), get_width(S2,W2),
    (Bottom1 #< Bottom2) #=> (W1 #< W2).
```

Note that constraints of this type are enabled by CLPFD's ability to post propositional constraints. The system found other such propositional constraints in partial scene 2, including the constraints that thinner rectangles are shorter and less saturated (and vice-versa). Note also that the system is able to detect global disjointness in a scene and derive an appropriate binary constraint for each pair of rectangles. It also determines the minimum amount of padding around each shape in the partial scene, and imposes this in the disjointness constraint, so that rectangles in the completed scene have similar padding.

2.2 Introducing Randomness

Unfortunately, the solutions to the kind of constraint problems which are generated tend to be fairly formulaic. For instance, the first scene presented in figure 2 is a valid solution to the constraints derived from partial scene 1 of figure 1. We see that the introduction of rectangles of the same shape, size and colour satisfies the constraints. In order to try to introduce more variety, such as that present in completed scene 2 of figure 1, we first experimented with an all-different constraint scheme and a constraint scheme which imposed a minimum difference between variables. However, we found that the former didn't introduce enough variety, and the latter tended to over-constrain the CSP. In addition, we also decided that for certain partial scenes, the user should be able to generate multiple

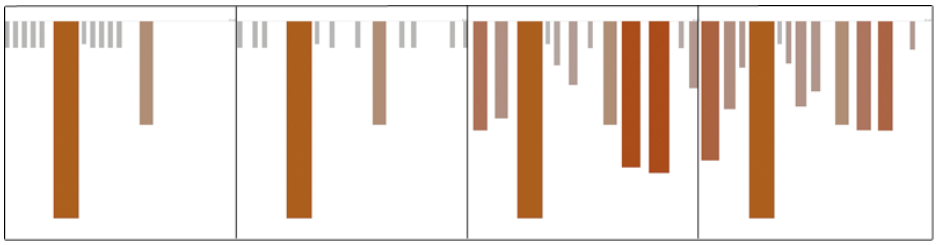


Fig. 2. Four completed scenes satisfying the constraints derived from partial scene 1 of figure 1. The first has no randomness imposed; the second has a random x-coordinate imposed and the final two have random x-coordinate, height and saturation imposed.

different scenes using the CSP. For these reasons, we allowed the user to specify that certain variables should have a random value ordering in the CSP search. As an example, to achieve the kinds of woodland scenes we desired from partial scene 2, we specified that the system should impose random value orderings on the height, saturation and x-coordinate of each rectangle. The completed scenes in figure 2 demonstrate the difference in visual variety achievable when the randomisation of variables is imposed. As described in section 3, we experimented to see whether reducing the domain of random values increased solving speed.

2.3 Using the CSP to Build Scenes

To recap, with our system, the user changes the shape, colour and location of a set of rectangles on screen, so that the partial scene exhibits the kinds of relationships they wish to see in the full scene. The system then induces certain constraints and the user can deselect any which are not there by design. Finally, the user chooses the number of additional rectangles, r, they wish to add, and the system is employed to complete the scene. We have experimented with two different ways to use the CLPFD solver to generate completed scenes. Firstly, we looked at trying to solve the scene generation problem in *one pass*. That is, a single CSP which contains variables for the shape, location and colour of r rectangles is written and solved. We impose a restart limit, which is useful when generating scenes with a random element. Secondly, we looked at a *sequential* approach where a constraint satisfaction problem is generated with variables for a single rectangle. This is solved and the rectangle is added to the scene, then the process iterates until the required number of rectangles are present. We impose a back-tracking parameter b: if a search for rectangle number n fails b times, then rectangle numbers $n-1$ and $n-2$ are removed from the scene, and the search starts again. When generating scenes with a random element, we found that over-constrainedness occurred when two rectangles with very similar values for the same variable were sequentially added to a scene. We further found that the scheme of removing both was more successful than removing just one of them, hence the reason for removing both rectangle $n-1$ and $n-2$.

3 Experimental Results

Our first set of experiments were designed to determine the fastest way to construct a completed scene from partial scene 1 of figure 1. We looked at both the one pass and the sequential approach, with the times taken to construct scenes with between 10 and 20 rectangles presented in figure 3. We see that, while the one-pass approach is faster for the addition of smaller numbers of rectangles, its solving time drastically increases when sixteen rectangles are required. Indeed, we found that constructing scenes with seventeen rectangles caused a memory problem which we have not yet overcome. In contrast, the sequential approach seems able to cope in a linear fashion with increasing numbers of rectangles, because it only has to solve a CSP for one rectangle at a time.

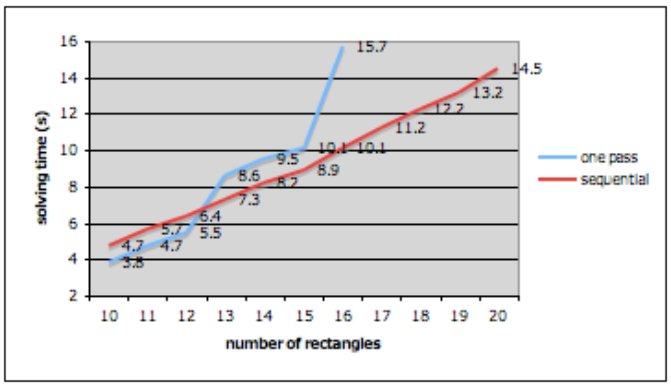


Fig. 3. Solving times for partial scene 1 for the one pass and sequential approaches.

Secondly, working with the constraints derived from partial scene 2 from figure 1, we experimented with various setups to determine the fastest way to construct a scene with twelve rectangles. In contrast to the first scene, scene 2 requires the x-coordinate, height and saturation variables to have a random value ordering, to achieve the desired visual variety. We experimented with different percentages for the size of the random variable domain, for instance, using a 17% random domain setting for a variable with integer domain d_1 to d_n means that a list of $0.17 * (d_n - d_1)$ values randomly chosen between d_1 and d_n without repetition becomes the value ordering for the variable. Using various settings for the backtracking search parameter and this domain percentage, we found that best time with the sequential approach was around 18 seconds. In contrast, we achieved much lower times for the one-pass approach. We experimented with various settings for the restart time (rs) and random domain percentage (rdp) and due to the random nature of the value orderings, for each pair $\langle rdp, rs \rangle$ which defined the search setup, we ran the system 10 times and recorded the average number of times the search was restarted and the average overall search time. The results are presented in table 1. We see that the lowest average time over a 10 run session was 2.6 seconds, which was achieved by both $\langle 20, 20 \rangle$ and

$\langle 15, 75 \rangle$ settings. Also, the average time over all the setups is 7.6 seconds, which is less than half the time taken with the best sequential approach. We also note that – on average – the random domain percentage which is fastest over all the restart times is 75%, and on average over all the random domain percentages, using a restart time of 3 seconds was the fastest.

	Random domain percentage																					
Restart (s)	2		5		10		15		20		25		30		50		75		100		av.	
2	8.0	20.4	3.5	9.1	2.4	6.3	1.3	3.4	1.2	3.1	1.4	3.7	1.2	3.1	2.0	5.3	1.2	3.1	1.7	4.6	2.4	6.2
3	4.0	13.7	1.5	4.5	1.5	4.5	1.6	4.8	1.5	4.4	1.4	4.0	1.4	4.0	1.3	3.7	1.8	5.5	1.4	4.0	1.7	5.3
5	2.7	13.7	1.9	8.2	1.4	4.9	1.2	3.8	1.8	7.4	1.4	4.8	1.5	5.5	1.1	3.2	1.4	4.0	1.8	7.0	1.6	6.3
10	1.6	11.6	1.2	4.9	1.3	7.6	1.2	5.6	1.3	6.6	1.4	7.7	1.2	6.3	1.5	10.5	1.4	6.9	1.5	8.8	1.4	7.7
15	1.6	20.2	1.5	12.5	1.3	8.7	1.3	8.6	1.3	8.9	1.6	14.7	1.4	11.6	1.5	11.8	1.0	**2.6**	1.4	11.5	1.4	11.1
20	1.2	12.1	1.0	11.7	1.0	8.1	1.0	9.9	1.0	**2.6**	1.0	6.3	1.0	8.0	1.0	13.6	1.0	6.3	1.0	8.2	1.0	8.7
av.	3.2	15.3	1.8	8.5	1.5	6.7	1.3	6.0	1.4	5.5	1.4	6.9	1.3	6.4	1.4	8.0	1.3	4.7	1.5	7.4	1.6	7.6

Table 1. Number of restarts and solving times (in seconds) for the one pass approach. The best times (of 2.6s) are shown in bold face.

We intend to perform much more experimentation over a larger set of partial scenes, in order to be more concrete about the best settings for certain scenes. Our preliminary conclusions are that, for the construction of larger scenes, the sequential approach is advisable, but for smaller scenes of 15 rectangles or less, the one-pass approach may be quicker. We cannot yet be conclusive about the best settings for generating scenes with random aspects, because the two best settings from table 1 were for relatively large restart times, but on average over the random domain percentages, lower restart times seem to be more efficient. Also, as random domain percentages vary, there seems to be no obvious trend in efficiency, with the exception of 2%, which is clearly too small.

4 Discussion

Recall that our ambition for The Painting Fool is for it to ultimately be taken seriously as a creative artist in its own right. We believe that to achieve this will require addressing as many misconceptions in the art world (and society in general) as it will require the solving of technical problems. In particular, in [4], we argue that the default position – that machines cannot be creative – can lead to a vicious circle whereby the value of machine generated art is assessed not in terms of the artwork itself, but in terms of the (seemingly uncreative) process which produced it. As a start to breaking this vicious circle, we propose that the authors of creative software describe how it operates in terms of high level notions, rather than in terms of its algorithms. In particular, in [4], we suggest that any software which needs to be considered creative in order for its products to be assessed favourably should be described in terms of its skill, appreciation and imagination (which we call the *creative tripod*). Moreover, we suggest that the development of creative software could be guided by the creative tripod, i.e., we determine whether our software needs more behaviours which could be described as skillful, appreciative or imaginative. For instance, until we worked with emotion detection software (as described in [5]), we could not describe The

Painting Fool as appreciative, whereas now we can claim that the software has a greater appreciation of both its subject matter and the way in which its painting choices can affect the emotional content of its pictures. We are currently working on implementing behaviours which could be described as imaginative, and we are concentrating on enabling The Painting Fool to invent novel scenes in a similar way to the AARON program. The scene generation approach described here forms part of this programme of work.

If one were to find a criticism of AARON [9] as a creative system in its own right, it would be that Cohen has neither given it any aesthetic preferences of its own, nor the ability to invent new aesthetic preferences. In this sense, AARON remains largely an avatar of its author. While this is actually the wish of the program's author [personal communication], a general criticism of software which we may purport to be creative, is that it never exceeds its training. To combat this, we advocate the practice of *climbing the meta-mountain*, whereby once a creative system exhibits one behaviour, we examine the next creative responsibility that is still held by the human user of the system, and move on to automate behaviours which take over that responsibility. At each stage, we advocate implementing the ability to exceed our training with respect to the behaviour just implemented. For example, for a project with The Painting Fool, which resulted in the Amelie's Progress Gallery (see `www.thepaintingfool.com`), we implemented a number of different artistic styles (comprising a colour palette, abstraction level, choice of natural media, paint stroke style, etc.), and used this to paint numerous portraits of the actress Audrey Tatou. In addition, we gave The Painting Fool the ability to invent its own artistic styles, by choosing aspects of the style randomly. While this was simplistic, it was effective: around half of the 222 portraits in the gallery came from randomly invented styles and the expert system of artistic styles mentioned above is composed in part with styles which were randomly generated. As another case study, as previously mentioned in [3], we employed a machine learning approach to invent fitness functions, and the resulting system could be perceived as more imaginative, because we were unsure in advance not only of what the generated scene would look like, but also what aesthetic (fitness function) the scene would be evolved to maximise. Again, we can claim that the combined system exceeded our training somewhat.

As with the notion of the creative tripod, the notion of climbing a meta-mountain provides a useful way of communicating ideas about creative software to a general audience: in this case, we are describing how we train software to emulate artistic abilities and exceed their training. However, the notion of climbing a meta-mountain can also guide the development of creative software. To this end, we are working to a seven point meta-level implementation scheme, whereby we add behaviours which simulate (a) making marks on paper (b) making marks to represent scenes (c) painting scenes stylistically (d) choosing appropriate styles for scenes (e) inventing scenes (f) inventing scenes for a reason (g) evolving as an artist. We are currently working on projects which lie around (e) to (f) in this scheme, and we acknowledge that latter aspects of the scheme – in particular (g) – are not well defined yet.

5 Conclusions and Further Work

Via an initial implementation and preliminary experiments, we have shown that constraint-based scene generation is feasible, in particular that partial scenes can be defined visually and desired relationships can be guaranteed. We ultimately envisage a scene generation system which uses a combination of case-based, evolutionary, constraint-based and other approaches to allow the most flexibility. There are serious questions about the generality of the constraint-based approach, and we may find that it is only appropriate for certain types of scenes. In addition to the testing of the constraint based approach on many more scene types, we also plan to implement more constraint derivation methods to enable the building of more types of scenes. Following our methodology of training the system and then implementing behaviours which exceed our training, we will investigate methods to automatically change constraints derived from a partial scene, and to invent wholly new constraint schemes for the generation of scenes. As with the combined system presented in [3], we hope that with these additional abilities, the perception of imaginative behaviour in the system will be heightened.

Acknowledgements

We would like to thank the anonymous reviewers of this paper, who provided some very interesting feedback. We would also like to thank the reviewer of [3], who suggested that scene descriptions could be derived from exemplars, which led to the work presented here.

References

1. M. Carlsson, G. Ottosson, and B. Carlson. An open-ended finite domain constraint solver. In *Proc. Prog. Languages: Implementations, Logics, and Programs*, 1997.
2. S Colton. *Automated Theory Formation in Pure Mathematics*. Springer, 2002.
3. S Colton. Automatic invention of fitness functions with application to scene generation. In *Proceedings of the EvoMusArt Workshop*, 2008.
4. S Colton. Creativity versus the perception of creativity in computational systems. In *Proceedings of the AAAI Spring Symp. on Creative Intelligent Systems*, 2008.
5. S Colton, M Valstar, and M Pantic. Emotionally aware automated portrait painting. In *Proceedings of the 3rd International Conference on Digital Interactive Media in Entertainment and Arts*, 2008.
6. M Dohmen. A survey of constraint satisfaction techniques for geometric modeling. *Computers and Graphics*, 19(6):831–845, 1995.
7. M Jamnik. *Mathematical Reasoning with Diagrams: From Intuition to Automation*. CSLI Press, 2001.
8. P Jiantao and K Ramani. On visual similarity based 2D drawing retrieval. *Journal of Computer Aided Design*, 38(3):249–259, 2006.
9. P McCorduck. *AARON's Code: Meta-Art, Artificial Intelligence, and the Work of Harold Cohen*. W.H. Freeman and Company, 1991.
10. D Waltz. Understanding line drawings of scenes with shadows. In Patrick Winston, editor, *The Psychology of Computer Vision*, 1975.

Computing Makes the "Man": Programmer Creativity and the Platform Technology of the Atari Video Computer System

Ian Bogost[1] and Nick Montfort[2]

[1] School of Literature Communication and Culture
Georgia Institute of Technology
686 Cherry Street, Atlanta GA 30332 USA
ibogost@gatech.edu
[2] Program in Writing and Humanistic Studies
Massachusetts Institute of Technology
77 Massachusetts Avenue, Cambridge, MA 02139 USA
nickm@nickm.com

Abstract. Some of the cultural and technical forces that influenced the creation of the "man" (the player-controlled element) in two early home video games, *Pitfall!* and *Yars' Revenge,* are discussed. We find that the specific nature of the Atari Video Computer System (also known as the Atari VCS and Atari 2600) as a computing platform enables and constrains what can be done on the system, and that it also encourages and discourages certain types of creative work. For these reasons, understanding the platform is essential to understanding the development of games on this influential system and, by extension, the early history of video games.

Key words: Platform studies, Atari VCS, Atari 2600, video game development.

1 Two Cases from the Early Days of the Video Game

Creativity intersects with computing in many ways — not only when computers are used to model creativity, but also when they are used to enable and constrain the creative work of developers of digital media. We examine two cases where programmers of the Atari Video Computer System (also called the Atari VCS or Atari 2600) created a "man," an in-game representation controlled by the player, in a way that was sensitive to the capabilities of computing platform being used. The same icon would not have been created on a system with different capabilities; this helps to show how the surrounding games would also not have been developed in the same way if a different game system was being targeted. The cases considered are those of David Crane creating Pitfall Harry for the Activision game *Pitfall!* and Howard Scott Warshaw creating the fly-like Yar for the Atari game *Yars' Revenge*. They represent

some of the interesting interactions between programming and platform that we have written about in our forthcoming book, which takes a platform studies approach to the Atari VCS [1].

The Atari VCS was an extremely influential early home video game system. It was neither the first home system nor even the first cartridge-based system, but its success was so great that in the early 1980s, "Atari" became synonymous with "video game system" just as "Coke" often means "soft drink" and "Google" generically means "search engine" today. The creative work done on the system had an influence on the development of video game genres and interface conventions, among other things.

The claim in our title that "computing makes the 'man'" should be taken in only the most literal sense. It simply means that these two "man" objects, Pitfall Harry and the Yar, are generated by a particular computing system running particular programs — they do not exist without computing. This claim is not supposed to mean that the technology of a platform determines the creative output of programmers, or determines the programmers themselves, in any simplistic way. The relationship between creativity and platform is a complex one. We hope that our discussion here will shed some light on this complexity and help to show how the particularities of a computing platform are involved in the process of creating digital media.

2 Pitfall Harry

Pitfall! is an important early platformer and a predecessor to the side scroller, a form of video game which was made famous by *Super Mario Bros*. In this form, the "man" is seen from the side and typically moves from left to right as the background and structures continuously appear on the right and disappear on the left. With its side view, the ability of Pitfall Harry, the game's hero, to jump, swing, and climb and fall between different levels, and with the need to drive this character horizontally toward treasures, *Pitfall!* managed to do many of the essential things that a side scroller did even though it didn't smoothly scroll its environment.

Pitfall! arose from a combination of influences, technical and cultural. It started with the challenge of creating realistically animating graphics on the Atari VCS. The sprites in early games were static — one unmoving graphic comprises *Combat*'s planes, *Slot Racer*'s cars, even *Superman*'s human characters. *Pitfall!* creator David Crane had already experimented with simple animation to great effect in *Grand Prix*, in which the cars have wheels with tire treads that spin at different rates depending on the car's speed. But he had previously sketched out an idea for a realistically moving man. This became the basis for Pitfall Harry.

Because of the limitations of RAM, ROM, and processor cycles that were inherent to VCS programming, graphics like sprites were not considered external assets that could be dropped into a game. VCS programmers used quad-ruled paper to sketch out designs for sprites, considering not only the 8-bit wide patterns needed to render a

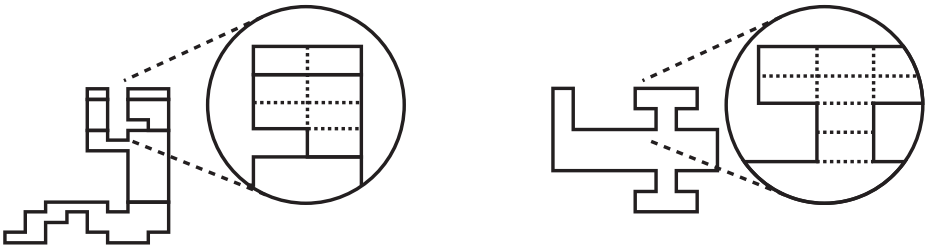


Figure 1. Pitfall Harry and the biplane from Combat are composed of the same divisions of the scan line, the wide pixels of the Atari VCS.

subject convincingly, but also how to design without changing sprite colors during a scan line and while accounting for the total size of a set of sprites in ROM. In some cases, the possible locations of a sprite on screen would dictate whether color changes were possible — for example, there might not be enough time to change sprite color and graphics values in addition to playfield graphics.

Another issue was the legibility of sprite graphics on-screen. The 8-bit width of VCS sprites do not provide a lot of room for detail, and some objects or creatures prove very difficult to render in such low resolution. Crane explained: "Early in my career at Atari, I designed a *Slot Machine* game. When I tried to draw traditional slot machine symbols — cherries, lemons, oranges, etc. — it became clear that there was no way to render those objects in 8 monochrome pixels. So I used cactus, cars and other angular objects that were easily recognizable when drawn with pixels" (Email to Bogost, 23 October 2007).

The smallest vertical unit in the VCS graphics system is a single scan line of the television screen. The smallest unit of horizontal space is a color clock, the amount of time it takes the machine's graphics chip, called Stella, to draw one color segment; Stella is configured to click away three color clocks for every cycle of the VCS's MOS Technologies 6502 processor. The detailed view of Pitfall Harry's head and neck drawn in Figure 1 on the left clearly shows that the "pixel" defined by these is rectangular, not square. This shape became a design feature; for example, the rectangular blocks help Pitfall Harry's legs appear longer than they would with square pixels. The *Combat* biplane sprite on the right appears to be comprised of square pixels because that program uses a two-line kernel, which only updates the sprite graphics every two scan lines.

The choice of the scorpion and cobra obstacles in *Pitfall!* evolved from a similar process, motivated more by how convincingly these opponents could be rendered than by any prior interest in those creatures.

Crane worked on the "little running man" animation for several months, refining its appearance and behavior. He walked deliberately around the office, trying to record his own leg and arm positions and to translate those movements onto pixel paper. However, Crane didn't do anything with the little running man right away. Each time

he finished a project, he would bring out the designs and think about a game that might make good use of it. Finally in 1982, a plan came together: "I sat down with a blank sheet of paper and drew a stick figure man in the center. I said, 'OK, I have a little running man… Let's put him on a path' (two more lines drawn on the paper). 'Where is the path?… Let's put it in a jungle' (draw some trees). 'Why is he running? … (draw treasures to collect, enemies to avoid, etc.) And *Pitfall!* was born. This entire process took about 10 minutes. About 1000 hours of programming later the game was complete.'" (Email to Bogost, 23 October 2007). The inspiration for *Pitfall!* wasn't the side-scrolling jungle adventure, but rather the running man. The adventure just gave him a reason to run.

Crane's technical innovation combined with several cultural influences to produce the "man" Pitfall Harry instead of just a generic running figure. Today, highly detailed videogame characters with complex backstories are common. Miyamoto's Jumpman (who later became Mario) and Iwatani's Pac-Man had become cultural icons before *Pitfall!* was released. But Pitfall Harry was the first popular video game character born on a home console system. He eventually spawned numerous sequels, licensed products, and even a television cartoon. The little running man was partly responsible, but the cultural references also helped to develop the game's fictional world.

The film *Raiders of the Lost Ark* was released in 1981. Crane acknowledges that the movie inspired the idea for an adventure in the jungle. But apart from that particular kind of wilderness setting and a guy who runs, little about the game resembles *Raiders*. (Howard Scott Warshaw's Atari-licensed Atari VCS *Raiders of the Lost Ark* cartridge takes considerable license with the film's character and the plot, but nevertheless has many more identifiable elements that can be read as related to the film.) Beyond the cinematic adventure of Indiana Jones, there were two important inspirations that contributed to Crane's design.

The first explains Pitfall Harry's ability to swing on a vine. This idea, of course, comes from Tarzan, the original vine-swinger, who was created by Edgar Rice Burroughs in 1912. Tarzan also inspired Taito's 1982 arcade game *Jungle Hunt*, although that game was developed independently of *Pitfall!*, with neither developer knowing about the other project. Perhaps jungle fever was in the air in that year.

The second explains the crocodiles in some of the *Pitfall!* ponds. From the 1940s through the mid-60s, Paul Terry's Terrytoons studio, best known for the character Mighty Mouse, released a theatrical cartoon series featuring two magpies named Heckle and Jeckle. The cartoons featured the typical amusing pranks, in which the two birds calmly outwitted a variety of foes. In one sequence, the two ran across the heads of crocodiles, deftly escaping their snapping jaws. Crane, who was born in the mid-1950s, remembered seeing the cartoons as a child. He speculated that this idea would make an interesting mechanic in an adventure game.

The result was interesting indeed, partly thanks to how it made the Heckle and Jeckle maneuver interactive. To the amateur player of *Pitfall!*, the screens with crocodile-filled ponds prove quite difficult. It is only possible to stand on the heads of

the crocs while their mouths are open, and a misstep lands Pitfall Harry in the water. As the player becomes more experienced, the player works up enough skill to jump quickly and deftly over the crocodiles, just like Heckle and Jeckle.

3 A Fly Named Yar

Howard Scott Warshaw's first assignment at Atari was the project that would eventually result in *Yars' Revenge*. Initially, he was to port the arcade game *Star Castle*, produced by Cinematronic, to the Atari VCS. As he told an interviewer, "I soon realized that a decent version couldn't be done, so I took what I thought were the top logical and geometric components of *Star Castle* and reorganized them in a way that would better suit the machine" [2]. Warshaw's comment reveals how the platform participates in the ecology of game development. The design of *Yars' Revenge* was not entirely determined by the Atari VCS, nor was it dropped on the platform by its programmer with no concern for how the system worked. The original idea was to imitate another game, but the capabilities and limitations of the VCS led the developer to create something different, a reorganization of *Star Castle*'s major components that recognized the differences between vector and raster graphics, exploited the abilities of the TIA, and was well-suited to home play.

It was a radical move for Atari to set aside *Star Castle*, which they had already arranged to license. An arcade game that was a hit would of course have a following already, one that might generate enthusiasm and an initial market. Even one that wasn't a huge success still contained a complete and fully implemented game design, one which had been tested on the playing (and paying) public.

Ironically, however, the hardware capabilities of an arcade machine — in terms of processing power, graphics, and controller setup — were always significantly different from those of the Atari VCS, so that having a well-tested and implemented game design implemented on an arcade platform didn't mean very much when it came to the home console's hardware. The most obvious difference between the underlying *Star Castle* computing system and the VCS was the arcade machine's vector graphics, which Atari called XY graphics. Atari's successful arcade games *Tempest*, *Battlezone*, *Asteroids*, and *Lunar Lander* all use this sort of graphics system, which employs a fundamentally different type of monitor. All early arcade games used a CRT, but the ones in vector graphics games are wired differently than are the ones in standard televisions. The electron beam does not sweep across the screen from left to right, being turned on and off as it makes its way down and then returns back to the top 60 times a second. Instead, the electron gun is pointed at a certain location, turned on, and moved from that (x, y) coordinate to another point in a straight line, where it is turned off again. (An oscilloscope functions in a similar way; it just uses a different method of deflecting the electron beam.) Because the beam can be made to move arbitrarily instead of progressively scanning along the screen, this way of

illuminating a phosphor-coated screen was also called "random scan."

Cinematronics was the first company to release an arcade game that used this display technology. The game was *Space Wars*, released in 1977. Like the first raster arcade game, Nolan Bushnell's pre-Atari *Computer Space*, it was a two-player game and an arcade implementation of the 1962 PDP–1 program *Spacewar*. *Star Castle* has significantly different gameplay, and was developed by a different person, but the space setting and control scheme show that it is clearly based on Cinematronics' earlier game *Space Wars*.

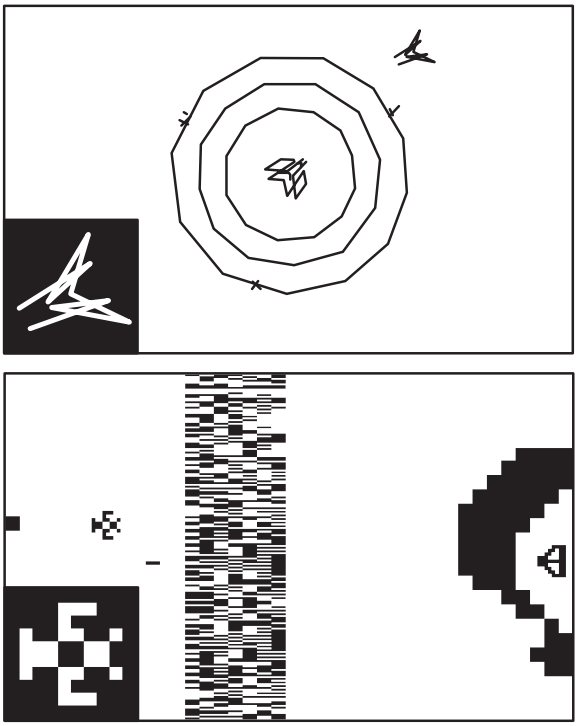


Figure 2. *Star Castle,* the vector-graphics arcade game shown at top, inspired *Yars' Revenge,* at bottom, which was made for the raster graphics of the Atari VCS. The *Star Castle* ship and *Yars' Revenge* "fly simulator," the Yar, are shown in detail in the lower left of each screen.

Vector graphics have certain advantages over raster graphics; at least, they did in the late 1970s and early 1980s. Specifically, it is much easier on a vector system to rotate shapes and to scale them up or down, as is needed when zooming in or out. For instance, to resize an object that is centered on (0,0) and consists of some set of lines, each endpoint's coordinates can simply be multiplied by the scaling factor. It does not get much trickier when the object is centered elsewhere or when there should be different scaling along the horizontal and vertical axes. It is also straightforward to shear a shape, keeping one axis constant while shifting the other. Rotation, scaling, and shear are the basic linear transformations, and any one can be accomplished by subjecting the coordinates of a shape's points to a single matrix multiplication. In combination with translation (the displacement of the shape in space, which just requires addition of the amount of the displacement), these allow for all the possible transformations that keep straight lines straight.

In a raster graphics system, particularly one with the limited computing power of the Atari VCS, the only feasible way to show a rotation of an object is to display a

different bitmap, a hard-coded image of the shape rotated by the desired amount. This is how tanks and planes in *Combat* are rotated, for instance. A simple form of scaling is supported in hardware, via the TIA's number-size registers, but smoother zooming has to be done by providing additional bitmap graphics. Even when the display hardware itself is not of the XY graphics sort, these benefits of vector graphics can be seen when comparing systems such as an early version of the bitmap-based Macromedia's *Director* and an early version of that company's vector-graphics *Flash* environment.

The object of *Star Castle* is to destroy the rotating cannon in the center of the screen repeatedly, with one's triangular, rotating ship, a vessel that looks and moves like the ones in *Asteroids* and *Space Wars*. The enemy cannon appears behind colored overlays and is surrounded by three concentric, rotating shields, each of which is made of line segments. The segments can be destroyed by fire from the player's ship, but whenever an entire ring is shot away, it regenerates. Whenever the player clears a path to the cannon, creating a chance to shoot at it to destroy it, the canon fires a large missile toward the player's ship. As the player is trying to break down the shield rings, three mines also move out and seek the player's ship. They can be avoided or shot, although shooting them does not increase the score and they soon reappear. After a central cannon is finally successfully destroyed, another one quickly appears with three intact rings around it.

In *Yars' Revenge*, the player's "ship" or "man" is the Yar, a "fly simulator" that is controlled with the joystick. *Yars' Revenge* does not have a ship that pivots clockwise or counterclockwise, something that was fairly easily implemented on the vector-graphics display system of *Star Castle*. Instead, the joystick is used to directly move the Yar in the standard eight directions — up, down, left, right, or diagonally. This is a form of movement that was fairly easy for the Atari VCS: translation while facing in one of eight directions. This type of mapping, a form of direct manipulation, wasn't exactly native to the system, though. Early games, including *Combat,* often used methods of joystick control that now seem far less obvious. The VCS cartridge *Asteroids,* a port of an arcade game that featured a rotating ship, used a rotate-and-thrust control scheme.

The Yar sprite is animated, requiring an additional frame for each direction, but its appearance facing right is a reflection of what it looks like facing left, allowing for some savings. Up/down reflection is not as straightforward as left/right reflection; the latter can be accomplished with a single write to a register on the VCS's graphics and sound interface board, Stella, while the former requires manually loading and storing new bitmap data. For this reason, the Yar sprites for up and down are both laid out in ROM. Switching between the two requires reading a different bitmap. The insect appearance of the Yar was simply based on what Warshaw could draw and animate in an interesting way within a player sprite. The name "Yar" has a more definite referent — it was devised by spelling Atari CEO Ray Kassar's first name backwards.

4 Standing on the Shoulders of Platforms

By choosing a platform, new media creators simplify development and delivery in many ways. Their work is supported and constrained by what this platform can do. Sometimes the influence is obvious: A monochrome platform can't display color, a video game console without a keyboard can't accept typed input. But there are many more subtle ways that platforms interact with creative production, due to the idioms of programming that a language supports or due to transistor-level decisions made in video and audio hardware. In addition to allowing certain developments and precluding others, platforms also encourage and discourage different sorts of expressive new media work with much more subtlety. In the case of *Pitfall!* and *Yars' Revenge*, the appearance, function, and behavior of each game's "man" was influenced by both cultural and technical factors. Pitfall Harry realized David Crane's attempt to realistically animate a running man. Crane used the platform's technical capabilities with virtuoso skill while also drawing on popular culture to create the character's context and program his behavior. The Yar was Warshaw's deft adaptation of a spaceship from a coin-op game with a totally different graphics system. The resulting figure went native on the home system, becoming an iconic character for the Atari VCS.

Particular platform studies may emphasize different technical or cultural aspects and draw on different critical and theoretical approaches, but to deal deeply with platforms and new media, these sorts of studies will all have to be technically rigorous. We hope our book-length consideration of the Atari VCS will be the first of many such studies. We have invited proposals for other books in the new Platform Studies series of The MIT Press, at <http://platformstudies.org>. We invite the consideration of all sorts of computing platforms, including not only video game consoles but also home computers of all sorts; important software systems that have been treated as platforms, from BASIC through HyperCard to Java; and minicomputer and mainframe systems such as PLATO and the PDP-10. The detailed analysis of hardware and code can connect to the experience of developers who created software for a platform and users who interacted with and will interact with programs on that platform. An approach that looks at a particular platform, or a closely related family of platforms, through time and in a variety of contexts of use is especially well-suited to uncovering new insights about digital media. The deep investigation of computing systems has the potential to reveal the interactions between platforms and creativity, design, expression, and culture.

Creators need to understand how the medium they work in produces meaning, aesthetics, and experiences. The same is true for the critic, who must be able, among other things, to grasp how an artifact signifies in relation to how it was created. Such principles apply to all expressive media, from paint to language to computation. Since the conventions and forms of media develop historically, it becomes increasingly important to understand the origins, trends, and alterations of creative expression. For

example, the modern poet benefits from an understanding of meter, which developed and changed throughout the millennia. Computational critics and creators should also be concerned with how form and material influence the way work is produced and received. Platform studies offers one inroad into such a practice, by focusing on the ways a platform's hardware and software interacts with processes of computational creation and reception.

References

[1] Montfort, N. & Bogost, I (2009). *Video Computer System: The Atari 2600 Platform*. Cambridge: MIT Press.

[2] Stilphen, S.. (2005). "Interview with Howard Scott Warshaw." April 23. <http://www.digitpress.com/archives/interview_warshaw.htm>.

Video Games Cited

Atari. (1979). *Asteroids*. Arcade. Developed by E. Logg and L. Rains.
Atari. (1981). *Asteroids*. Atari VCS. Programmed by B. Stewart.
Atari. (1980). *Battlezone*. Arcade. Programmed by E. Rotberg.
Atari. (1987). *Breakout*. Atari VCS. Programmed by B. Stewart.
Atari. (1977). *Combat*. Atari VCS. Programmed by J. Decuir and L. Wagner.
Atari. (1978). *Slot Racer*. Atari VCS. Programmed by W. Robinett.
Atari. (1979). *Superman*. Atari VCS. Programmed by J. Dunn.
Atari. (1980). *Tempest*. Arcade. Programmed by D. Theurer.
Atari. (1979). *Lunar Lander*. Arcade.
Atari. (1982). *Raiders of the Lost Ark*. Atari VCS. Programmed by H. S. Warshaw.
Atari. (1981). *Yars' Revenge*. Atari VCS. Programmed by H. S. Warshaw.
Bushnell, N. (1971). *Computer Space*. Arcade. Distributed by Nutting Associates.
Cinematronics. (1977). *Star Castle*. Arcade.
Cinematronics. (1977). *Space Wars*. Arcade.
Crane, D. (1982). *Pitfall!* Atari VCS. Activision,
Crane, D. (1982). *Grand Prix*. Atari VCS. Activision,
Nintendo. (1985). *Super Mario Bros*. NES. Designed by S. Miyamoto.
Taito. (1982). *Jungle Hunt*. Distributed by Midway.

Uninformed resource creation for humour simulation

Graeme Ritchie

Computing Science
University of Aberdeen
Aberdeen AB24 3UE, UK.

Abstract: When testing a model of humour generation, it may not be feasible to implement certain crucial modules. We outline how human knowledge manipulation could be used to simulate these modules, without undermining the rigour of the testing process.

Keywords: humour generation, methodology, annotation.

1 Introduction

The area of computer generation of humour [1, Chap. 10] aims to simulate the creation of humorous artefacts (usually written texts), typically via a number of processing stages. If a humour-generating program is needed for some practical purpose, then how it is constructed may be of little theoretical interest. However, in the case of more theoretical research, where the program is there to test an abstract model, the mechanisms are the whole point.

A decomposition of the humour-generation process could lead to various situations. It could be that all steps can be computed with resources which exist or can be computed automatically (e.g. [2]). Or some of the steps might be impossible to compute because they involve substantial unsolved problems. Here, we consider a third possible situation: where some of the steps are relatively unproblematic theoretically, but happen to be difficult or impossible to compute, for practical reasons not relevant to the simulation of humour. The solution proposed here is to use human intuition to bridge these gaps, but in a way which does not vitiate the overall model of humour. Normally human intervention in a computational model of a 'creative' process is seen as invalidating the testing of the model, but we suggest here that a suitably controlled and constrained human contribution can be wholly valid.

2 Uninformed Resource Creation

In other areas, it is routine practice to use human judgements to create data resources. Computational linguists make much use of annotated corpora, such as the Penn Treebank [3], where some quantity of language data has been marked up by human judges. Sometimes known experts are used for these tasks, but sometimes the work is done by relatively untrained persons working from explicit guidelines. Standard methodologies exist, involving (for example) checks

on inter-rater agreement [4]. Often, the annotation is not specialised to some very specific subsequent use – the aim is to build a general-purpose resource, usable in whatever way other researchers see fit (not necessarily for text generation).

Similarly, in experimental psychology, it is well-established that materials to be used in an experiment may need to be pre-processed, using human reactions or judgements, to 'norm' or to 'validate' them. For example, van der Sluis and Mellish [5], in a test of the effects of 'positive' and 'negative' texts on readers, used a first stage to establish which sample texts were actually 'positive'/'negative'. Such a preliminary validation step is not part of the actual experiment, but merely calibrates the data. In such cases, the subjects may be given some explicit remit (e.g. guidelines) about how to classify the data, but quite often the calibration will be carried out by measuring reactions or effects. Such validations are relatively specialised for the particular experimental setting, even though the resource could be used elsewhere if it was suitable.

In both data annotation by specialists, and norming of data by untrained subjects, the human judge's knowledge or reactions are being used for a task which it would not be feasible to automate. However, the actual process of annotation or validation is not itself intended as a component in some wider model – it is just a practical way to create the resource. These are the direct antecedents of the methodology being proposed here.

The idea is that where a humour model requires some stage which is currently not computable automatically, the researchers should formulate an objective, explicit, modular, detailed specification of the stage, without reference to its place in the model, especially with no reference to humour. Then, using that specification, human volunteers should create, using their own judgements (and possibly existing data resources) a resource which will allow the computation of that step. This whole procedure may itself have substeps which have to be treated in the same way (a recursive application of the methodology).

In this way, the researchers can develop a step-by-step simulation of the complete humour generation process. Some of the steps will have involved human intervention and intuition – but that has occurred in isolation, 'off-line', without reference to the humour model or to humour as the purpose, and so does not undermine the claim to have a mechanistic model. The term 'uninformed' is used here to emphasise that those humans who are applying their intuition are not aware of the purpose or theoretical context of their efforts, and are not directing their intuition to the central problems of the research. In particular, steps should be taken to minimise the chance that the participants will provide (or try to provide) data which is itself 'funny' (even without the other modules in the process). In developing a theoretical account of the underlying mechanisms of humour, it is important that the overall phenomenon is decomposed into – and hence explained in terms of – components which *are not themselves humorous*. If a model of humour contained a module whose remit was 'find a funny idea', then it would be hard to argue that the model gave rise to humour – the humour would have been supplied ready-made by this single module (and the question would then be: what is inside this module to make that happen?).

3 How It Could Work

A very simple example of this approach is the way that the first prototype of the JAPE riddle generator [6] used a lexicon solicited from volunteers. However, to illustrate how a component in a processing model could be handled, consider Binsted and Ritchie's (unimplemented) model of generating 'story puns' [7]. A story pun consists of a short narrative (the setup) followed by a single-sentence (or single-phrase) punchline which is phonetically similar to some well-known phrase or saying (e.g. a proverb or quotation) and which somehow acts as a suitable conclusion to the story (often by summing up the message of the story). Binsted and Ritchie limit themselves to the case where the well-known saying is a proverb, and offer these five stages[1]:

(i) choose a maxim;
(ii) distort that maxim;
(iii) construct a semantic analysis of the distorted version;
(iv) derive constraints from that semantic form;
(v) devise a story to meet these constraints.

The first two stages should be open to automation using a collection of proverbs and some information about phonetic similarity. The third stage is in principle automatable, but difficult in the current state of the art, and the fourth stage hard to define formally. The fifth stage - story generation - is clearly challenging. To illustrate our approach, we suggest replacing stages (iii), (iv), (v) with:

(iii′) paraphrase the distorted maxim in a way that uses a different phrasing;
(iv′) given a sentence offering general advice, outline a sequence of events which would exemplify the truth of that advice.
(v′) given a sequence of events, write a narrative text describing them.

In a sense, this uses the paraphrase in place of a formal semantic representation, since this facilitates its use in guiding the story generation. This new stage (iii′) might be computable automatically, but if not, human participants could be briefed to carry out this task. Stage (iv′) would start from the results of (iii′). This could not be automated at the moment, but it could – if suitably specified – be performed by humans. That is, stage (iv′) could be computed offline by a participant who was told to list a situation, an event, or a sequence of events, which illustrated the wisdom of the advice in the supplied sentence. Similarly, (v′) could be delegated to human volunteers. It might be better to merge stages (iv′) and (v′), asking the volunteers to write short stories to illustrate the applicability of the advice, rather than having the two stages of stating an abstract spine for the story (stage (iv′)) and then having the story itself created ((v′)). Appending the distorted adage (which the stage (iv′)/(v′) volunteers have never seen) to the story would complete the construction of the story pun.

It would also be possible to have a further validation phase (for any of the stages), as in Binsted's lexicon creation, where different judges vet the data and sift out potentially faulty items.

[1] There is also an implicit sixth stage: append the result of (ii) to the result of (v).

Once again, the processing has been 'de-skilled', in the sense that non-humorous mappings (paraphrasing, story-writing) have been executed by human craft without subtler judgements about humour or the overall model.

4 Summing Up

For this approach to be viable, the humour model must have well-defined components, and any component being 'simulated' by humans must meet these criteria:

- The researcher must be able to specify clearly, objectively, and in sufficient detail what the functionality of the component is.
- The component must be modular, in order to be treated as a isolated task.
- The nature of the component must not encapsulate the essence of humour (e.g. the devising of a 'funny' text); see Section 2.
- The researcher must be able to specify the computation of that component in terms intelligible to suitable human participants who have no knowledge of the broader theoretical context.
- The specification for the participants must not reveal how the resource they create will be used – the task must have a self-contained description.

This methodological note argues that even if some component of a humour-creation model is not yet fully automatable, the model can still be tested, and a test involving human input need not be dismissed as invalid. This perspective might also be applicable in other areas of computational creativity.

Acknowledgements: Thanks to the Aberdeen NLG group for discussion of these ideas. The writing of this paper was partly supported by grant EP/E011764/1 from the UK Engineering and Physical Sciences Research Council.

References

1. Ritchie, G.: The Linguistic Analysis of Jokes. Routledge, London (2004)
2. Stock, O., Strapparava, C.: The act of creating humorous acronyms. Applied Artificial Intelligence **19**(2) (2005) 137–151
3. Marcus, M.P., Santorini, B., Marcinkiewicz, M.A.: Building a large annotated corpus of English: The Penn Treebank. Computational Linguistics **19**(2) (June 1993) 313–330
4. Carletta, J.: Assessing agreement on classification tasks: the kappa statistic. Computational Linguistics **22**(2) (1996) 249–254
5. van der Sluis, I., Mellish, C.: Towards affective natural language generation: Empirical investigations. In Mellish, C., ed.: Proceedings of the Symposium on Affective Language in Human and Machine, AISB 2008, Society for the Study of Artificial Intelligence and Simulation of Behaviour (2008) 9–16
6. Binsted, K., Ritchie, G.: Computational rules for generating punning riddles. Humor: International Journal of Humor Research **10**(1) (1997) 25–76
7. Binsted, K., Ritchie, G.: Towards a model of story puns. Humor: International Journal of Humor Research **14**(3) (2001) 275–292

What Happens Next?: Toward an Empirical Investigation of Improvisational Theatre

Brian Magerko and Mark Riedl

Adaptive Digital Media Lab
Georgia Institute of Technology
School of Literature, Communication & Culture
686 Cherry Street, Atlanta, GA 30332-0165
magerko@gatech.edu, riedl@cc.gatech.edu

Abstract. We are engaging in the experimental study of improvisational actors in order to fully comprehend and computationally model what actors do during improvisational performances. Our goal is to fill in the gaps of our understanding of creativity and improvisation and to begin to integrate engineering methods with the theory and practice of acting. This paper presents the current methodological issues of a new experimental study of cognition in improvisational acting.

Keywords: improvisation, theatre, cognitive modeling, methodology, creativity

Introduction

Improvisation in a theatrical performance can occur on many levels depending on the genre and style of performance, from the displaying of emotion, gesture, and posture, to the real-time determination of dialogue and narrative progression. There is a spectrum of improvisation in theatrical acting, from completely predetermined to completely non-predetermined [1]. On one side of the spectrum is the extreme case in which the performance of actors is completely scripted – displays of emotion, gesture, posture, positioning, dialogue, and verbal inflection and tone are all predetermined such that there is no tolerance for variation or individual interpretation. A more common model of theatrical performance is one in which dialogue and some instructions to actors are scripted, but the actors are responsible for interpretation and execution of the details of the performance. Differing styles of performance and interpretation of scripts involve differing degrees of improvisation skill.

Towards the opposite end of the spectrum one will find the genre of theatre called *improv* [2; 3]. Improv is a remarkable example of creative group performance because (a) the creation process of narrative content is completely in real-time, (b) there is no explicit coordination between the actors, and (c) in the case of unscripted improv, the constraints on a performance are typically of the form of a set of game rules plus audience suggestions. At the extreme other end of the spectrum is a form of improvisational performance in which there are no constraints on the activities of the actors given a priori, and everything is fabricated in situ.

There are schools of improv that have resulted in seminal publications and teaching methods (e.g. Johnstone's [2] teachings on "status transactions" and storytelling in improvisation or Spolin's [3] theatre games, such as the game "What Happens Next?"). These schools have elicited specific domain knowledge that expert improvisers are well versed in (e.g. strategies like "always accepting what is offered to you" or easy methods of establishing character, like Johnstone's "Fast Food Laban" and "Fast Food Stanislavski"). There have been attempts to codify some of this knowledge in the form of computational systems (e.g. [4]). However, the representation of these teachings can only bring us so far in our understanding of creativity. Comprehending how improvisation is taught (which has not been fully represented in a computational system to date anyhow) does not further our understanding of what actors *actually do* when improvising. To reach that level of knowledge, we need engage in the experimental study of actors performing improvisational acts. This approach has the potential to elicit deeper knowledge about what cognitive processes are being engaged during an improvised creative act (as opposed to representing concepts taught in seminal pedagogical texts).

Our goal is to fill in the gaps of our understanding of creativity and improvisation and also begin to integrate engineering scientific methods with the theory and practice of acting. Our key objectives are:

- The development of appropriate methodologies for collaborating with actors in order to capture data relevant to cognitive modeling. Our current work has focused on methodologies from (a) observational study [5] to (b) verbal protocols to (c) retrospective interview.
- The identification of (a) cognitive processes underlying improvisation, and (b) declarative and procedural knowledge employed in improvisation through a variety of experiments involving human actors.
- The development of implementable human behavior models based on our findings that contribute to research on autonomous agents and virtual humans that are capable of demonstrating creativity in the context of improvisational acting with or without human interactors.

The results of this work will be directly applicable to the development of autonomous agent and virtual human technologies that can improvise performances or act in the presence of human interactors. One of the intended outcomes of this ongoing work is the advancement of the design of autonomous agents for entertainment, distributed education, and training that are capable of reasoning about themselves, other autonomous agents, and human interactors to perform believably and effectively in uncertain environments where real-time behavior is essential. This paper presents the current methodological issues of a recently begun experimental study of cognition in improvisational acting and the application of the resulting theory to the behaviors of synthetic characters.

Improvisation

Theatre has been suggested as a model for believable agents [4; 6; 7]. A believable agent is an autonomous agent capable of expressing emotions, personality, and other

visual and behavioral qualities that facilitate the willing suspension of disbelief. However, little has been done to investigate the creative processes that underlie improvisation in theatre. To increase our understanding of creativity in the context of improvisation in theatre, and our understanding of improvisation and creativity in general, our research investigates (a) new methodologies for studying improvisation (b) new computational models of creativity, emotion, and improvisation in the context of theatrical acting, and (c) how to apply these models to the performance of believable characters.

Improvisation in the arts (i.e. theatre, dance, and music performance) can be differentiated from other domains in which improvised problem solving occurs because actors select actions for creative purposes rather than solely ones of correctness or efficiency (e.g. improvisation is a means for handling unexpected failures in domains such as organizational management and emergency response). We view improvisation as a group activity that involves communication between the performers as well as an array of cognitive processes to formulate and select actions quickly.

The current body of research on improvisation, which most notably comes from the improvisational music domain, points to the following generalities:

Improvisation is a constant process of receiving new inputs and producing new outputs [8; 9]. Improvisational dance, theatre, music, etc. all depend on performers observing their own and other performers' actions, performing some quick deliberative process, and then selecting new actions to perform. An improvisation model must be able to process and interpret these inputs as knowledge involved in decision-making.

Improvisation is a "continuous and serial process" [10] as opposed to one that is "discontinuous and involving iteration," such as music composition [11]. This suggests that there are specific cognitive processes that are employed during improvisation that are either a) different from those used during non-improvisational acts or b) are used with different constraints than those used during non-improvisational acts.

Improvisation involves decision-making based on domain-specific as well as real-world knowledge [9; 12; 13]. A key aspect of this knowledge is the use of a "referent" as background knowledge for improvisation [8; 10; 11]. A referent eases cognitive load during improvisation by providing material for variation, allows for a palette of pre-performance structures to be created by the performer, and reduces the need for complex communication cues with other performers. Therefore, a model of improvisation should be capable of processing and applying these kinds of semantic structures for both inspiration and the lessening of cognitive load.

Collaborative improvised pieces (as opposed to solo works) may involve both explicit and implicit communication [9; 14]. Body language, domain-specific cues, and verbal commands all contribute to the collaborative process of performing a group act. Any model of improvisation needs to address how communication to others in the group is used for coordination.

Improvisation is a process of severely constrained human information processing and action [8; 12; 15]. As Pressing [8] points out, an improviser must, in real-time, optimally allocate attention, interpret events, make decisions about current and future actions, predict the actions of others, store and recall memory elements, correct errors,

control physical movements, and integrate these processes seamlessly into a performance. How this view of cognitive constraints maps on to the theatre improv domain has yet to be shown.

Toward a Methodology for Studying Improv Actors

As described earlier, we are designing experiments with improvisational actors to elicit data on the cognitive processes and knowledge structures that they employ in order to build computational models of improvisational behavior. Examples of the hypothesized knowledge used include cultural knowledge, dramatic principles, practical experience, heuristics for selecting actions based on previous events, and meta-cognitive processes such as self-monitoring to stay "in character" and intentions of collaborators. Our current task is to design and execute an experimental design. Unfortunately, studying improvisers is an inherently difficult task. Direct observation of improvisational acting, such as MRI studies or traditional think aloud protocols, are impractical; performers cannot group together in an MRI machine nor can they think aloud while they are performing typical improvisation games. This forces us to consider novel methods for studying performance that allow us to get as much useful data as possible while at the same time having little to no effect on the participants as they perform.

Our first problem to solve is: *how do we collect data?* There are previous observational methods from social psychology and ethnography that can be adopted. Our design consists of observing actors as they improvise and coding the observed behaviors [1]. Our initial coding dimensions are:

- Referent codes (e.g. using game rules)
- Standard improv techniques (e.g. rules from Johnstone [2] or Spolin [3])
- Coordination (e.g. dictating other actor's character, establishing platform, (i.e. the setting, characters, and relationships in the scene))
- Workload (e.g. how long it takes to enter a scene, how long it takes to respond to a question or command, errors that occur, storing & recalling memory, etc.)
- Action selection / generation (e.g. heuristics for selection, generation of actions to consider, execution of selected action)
- Intention (actor's intentions and their model of the other actor's intentions)
- Error correction (e.g. recovery from misunderstanding a coordination attempt)

We will apply codes to videotaped improvised performances within an experimental setting based on both experimenter observation and retrospective protocol collection. The observational data will rely on our understanding of cognitive psychology and improvisational acting to create an appropriate coding scheme, as suggested by the dimensions proposed above. The retrospective protocols will involve showing performers a videotape of a performance that they have just created and continuously guiding them to comment on what they were trying to do.

Our second problem is: *what data do we collect?* Our current design is geared towards presenting players with modifications of typical improv games that are designed to elicit data of a specific nature. We are exploring methods for both *within-game* modifications as well as *meta-game* modifications.

Within-game modifications can be viewed as different dimensions that define a game that can be mapped to the different coding dimensions described above. These modifications change how the actor(s) improvise a scene, very similar to how improv game rules dictate how the scene is played (e.g. "you are only allowed to speak in questions"). Current within-game dimensions include:

- Amount & kind of game constraints: How specific are the game rules? How much of the platform is given? How much of the narrative is dictated?
- Response time: How much time is given to respond to their other actors? What amount of preprocessing time is given before a game begins?
- Privilege: Who is given what constraints? (e.g., one actor could be given an entire platform in secret while another is given none).
- Number of actors on stage
- Length of scene
- Narrative control: Who is told to lead the scene, if anyone?

Meta-game modifications deal with how each game is run by the experimenter. For example, we will run the same scene multiple times with different actors. This will give us a rich data set across individual and group dynamic differences. We will disrupt scenes to effectively ask, "what are you thinking?" to get one-time direct observations of actor cognition. This is not so different from certain improv game rules (e.g. the games "What Happens Next?" or "Freeze").

The specific choice of games will also be a meta-level issue in experimentation. Given to the large amount of data these experiments will produce, we are aiming for shorter scenes when possible and simpler games using dyads or triads of actors instead of entire troupes. Further we are considering reducing complexity by limiting the communication modalities that actors have available to spoken word. If we can see evidence of decision-making, collaboration, workload issues, etc. in these sparse performances, then we will have a strong story to tell about fundamental cognitive issues in improvisational theatre performance.

Toward Computational Modeling of Improv Actors

Once we have constructed a theory of cognition in improv from our data, we need to build a computational model based on it for evaluation. The next stage, therefore, is to implement the built theory in a computational architecture. We will implement the codified theory as declarative and procedural knowledge in a Soar agent [16]. The purpose of this is threefold. First, since the Soar architecture is a well-established, independently researched theory of human cognition, the extent to which our informal model maps into Soar formalisms tests whether we have considered the requisite aspects of cognition to build a human behavioral model. The Soar architecture provides many of the cognitively motivated mechanisms (e.g. declarative and procedural memory, heuristic selection of actions, etc.) that allow us to build on a pre-existing unified theory of cognition, rather than creating a model from scratch. Second, this enables our investigation to contribute to research on intelligent agents because the formalization of our initial model of improvisational acting doubles as a potential approach to the design of autonomous agents that are situated in uncertain,

unpredictable, real-time environments. Third, Soar has been effectively used to build large-scale interactive agents [17; 18]. Part of this implementation will include a meta-analysis of the suitability of Soar, which will hopefully identify future architectural needs as well.

Works Cited

1. Courtney, R. (1973). Theater and Spontaneity. Journal of Aesthetics and Art Criticism, 32(1).
2. Johnstone, K. (1981). *Impro: Improvisation and the Theatre*. Routledge / Theatre Arts, New York.
3. Spolin, V. (1963). *Improvisation for the theater; a handbook of teaching and directing techniques*. Evanston, IL: Northwestern University Press.
4. Hayes-Roth, B. and van Gent, R. (1996). *Story-Making with Improvisational Puppets and Actors*. Technical Report KSL-96-09, Knowledge Systems Laboratory, Stanford University, Palo Alto, CA: Stanford University.
5. Bakeman, R., & Gottman, J. M. (1997). Observing interaction: An introduction to sequential analysis (2nd ed.). New York: Cambridge University Press.
6. Goldberg, A. (1997). Improv: A system for real-time animation of behavioral-based interactive synthetic actors. In R. Trappl and P. Petta (eds.) *Creating Personalities for Synthetic Actors* (pp. 58-730). Springer-Verlag.
7. Perlin, K. and Goldberg, A. (1996). Improv: A System for Scripting Interactive Actors in Virtual Worlds. *Proceedings of SIGGRAPH'96*.
8. Pressing, J. (1998). Psychological Constraints on Improvisation. In B. Nettl and M. Russell (Eds.) *In the Course of Performance, Studies in the World of Musical Improvisation*, 47-67.
9. Reinholdsson, R. (1998). Making Music Together: An Interactionist Perspective on Small-Group Performance in Jazz. *Acta Universitatis Upsaliensis: Studia Musicologica Upsaliensia*, Nova Series 14.
10. Mendonça, D. and Wallace, W. (2007). A Cognitive Model of Improvisation in Emergency Management. *IEEE Transactions on Systems, Man and Cybernetics – Part A: Systems and Humans*, 37(4), pp.547-561.
11. Sarath, E. (1996). A New Look at Improvisation. *Journal of Music Theory*, 40(1), pp. 1-38.
12. Weick, K.E. (1998). Introductory Essay: Improvisation as a Mindset for Organizational Analysis. *Organization Science*, 9(5), pp. 543-555.
13. Yokochi, S. and Okada, T. (2005). Creative Cognitive Process of Art Making: A Field Study of a Traditional Chinese Ink Painter. *Creativity Research Journal*, 17 (2&3), pp. 241-255.
14. Alterhaug, Bjorn. (2004). Improvisation on a Triple Theme: Creativity, Jazz Improvisation, and Communication. *Studia Musicologica Norvegica*, vol, 30, pp. 97-118.
15. Johnson-Laird, P.N. (2002). How Jazz Musicians Improvise. *Music Perception*, 19(3).
16. Newell, A. (1990). *Unified Theories of Cognition*. Harvard University Press, Cambridge, MA.
17. Laird, J. E., Jones, R. M., & Nielsen, P. E. (1998). Lessons learned from TacAir-Soar in STOW-97. *Proceedings of the Seventh Conference on Computer Generated Forces and Behavioral Representation*. Orlando, FL.
18. Magerko, B., Laird, J.E., Assanie, M., Kerfoot, A., and Stokes, D. (2004). AI Characters and Directors for Interactive Computer Games. *16th Innovative Applications of Artificial Intelligence Conference*, San Jose, California, 877-883.

A Computer Model for Novel Arrangements of Furniture.

[1]Alfredo Aguilar[†], Diego Hernández[†], Rafael Pérez y Pérez [*], Mirelli Rojas[†], Ma. de Lourdes Zambrano[†]

[*]Departamento de Tecnologías de la Información
Universidad Autónoma Metropolitana, Unidad Cuajimalpa
México D. F.
rpyp@rafaelperezyperez.com
[†]Posgrado en Ciencias e Ingeniería de la Computación
Universidad Nacional Autónoma de México
e-mails: aah_sic@hotmail.com, diego_kike89@yahoo.com.mx, mirojas_1910@hotmail.com, malozaru@hotmail.com

Abstract. This paper reports a computer program that generates novel designs of arrangements of furniture within a simulated room. Although it is based on the E-R computer model of creativity, the current prototype only implements the engagement part. In this document we explain the core parts of the program and how we build our knowledge structures. We present some experiments, conclude that our program is able to generate novel arrangements of furniture, and describe some future work.

Keywords: *engagement-reflection, designs, emotional reactions, tension, context, atom.*

1 Introduction

The spatial arrangement of furniture within a room influences individuals' behaviour and emotional reactions. For example, people tend to classify furniture as belonging to specific areas within a house. In this way, when we walk into a bedroom we expect to find a bed, a television, a closet, probably a desk, etc.; all these elements belong to the set of bedroom-furniture and therefore we can think of them as establishing what we refer to as a harmonic relationship. In the same way, we might be surprised (an emotional reaction) if we find within a bedroom a washing machine. Since the washing machine does not belong to the set of bedroom-furniture, the harmonic relationship between the elements inside the room is broken leading to an individual's emotional reaction (e.g. surprise, stress). The elements that comprise the set of bedroom-furniture, or the set of kitchen-furniture, etc., and the types and intensity of individuals' emotional reactions depend on cultural, traditional and economical aspects.
There are several approaches to the research of the relation between emotions, behaviour and physical spaces. For example, Pullman and Gross [1] studies how to create emotional nexus with guests or customers through careful planning of tangible and intangible service elements within a business; what they are interested about is in analyzing emotional responses as mediating factors between the physical and relational elements that conform customers' experience, and their loyalty behaviours. Ritterfeld [2] points out the importance of the functional value (being used for specific purposes) of quotidian objects as an essential part of the process of aesthetic impression formation in daily life. Mehrabian and Diamond [3] studies how specific arrangements of chairs influence the way a conversation takes place. Lang [4]

[1] Authors are listed in alphabetical order.

studied how teachers in high-school alter the classroom-space in order to achieve their pedagogical goals.
We have developed a computer model able to represent emotional reactions to a specific arrangement of furniture within a room and then, following the engagement-reflection (E-R) computer model of creativity [5], employ this emotional representation to guide the generation of novel arrangements of furniture. The E-R model has been successfully employed to develop a computer program for plot generation known as MEXICA. Its main characteristics are:

- The user provides a set of examples that are employed to build the system's knowledge structures.
- The model includes two core processes: a generation process (known as engagement) and an evaluation processes (known as reflection). During engagement the system generates material guided by content constraints; during reflection the system evaluates the material generated so far —modifies it if it is necessary— and as a result of the evaluation adjusts the constraints that drive the generation of material during engagement. The system's outputs are the result of the interaction between engagement and reflection.
- An important characteristic of the model is that, following Gelenter [6], emotions are the glue that link ideas during engagement. Thus, in MEXICA, emotional links and tensions between characters drive the generation of material during the unraveling of the plot [7].

The main purpose of this work is to explore if the ideas that inform the E-R model employed for building MEXICA can be applied in interior design (see Alvarado & Pérez y Pérez in this proceedings for an example of the use of the model in music; see Acosta & Pérez y Pérez [8] for an example of the use of the model in creative problem solving). We believe that the study of the commonalities and differences in the use of the E-R model in different domains might result in interesting information about computational creativity. The program we report on this paper only implements the engagement part of the E-R model (we are in the process of designing the reflection part). It has two possible operation modes: user-mode and design-mode. The user-mode provides an interface that represents a room and ten pieces of furniture. The user can employ the interface to generate several designs of arrangements of furniture. The system records this information to create its knowledge-base. The design-mode generates automatically a novel arrangement of furniture. In this work, a novel arrangement of furniture means a design that is not recorded in the system's knowledge-base. It is out of the scope of this paper to perform any study related to individual's emotional reactions to spatial arrangements of furniture. Therefore, for the present version of the program, we intuitively define a set of rules that establish three types of possible reactions. However, our model is able to deal with a wider variety. In the following lines we describe the room and household goods we work with, how we represent emotional reactions and how the user and design operation modes work.

2 Description of the Room.

The current version of program works with the representation of a single-room (see figure 1). The room is defined as four walls surrounding an area of 3x3 meters. We identify two different types of positions within the room:
Centres. The room has five centres: the positions located at the centres of wall-1, wall-2, wall-3 and wall-4 are represented by the symbols c1, c2, c3 and c4 respectively; the centre of the room is represented by the symbol c.
Corners. The four corners in the room are represented by the symbols e1, e2, e3 and e4.

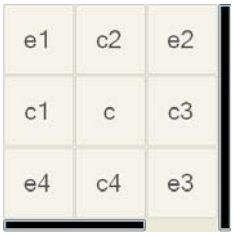


Fig. 1 Representation of a room.

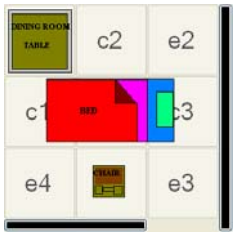


Fig. 2 An example of arrangement.

The current version of our program includes a list of ten possible pieces of furniture to be employed: a bed, a dining-room table, four single chairs, a vase (which is considered as one piece of furniture), a couch, an armchair, and a small-coffee table. Thus, it is possible to locate the table on e1, the bed on c and the chair on c4. The user can select the direction that each piece of furniture has; e.g., the bed can be located on c pointing towards the east (see Fig. 2).

3 Description of Tensions and Knowledge Structures.

We refer to the representation of emotional reactions to an arrangement of furniture as tensions. We distinguish three types of tensions: *Tension of Utility (Tu)*, *Tension of Proximity (Tp)* and *Tension of Distribution (Td)*.

Tension of Utility (Tu). In our model, furniture is designed to serve a purpose. For example, the purpose of a chair is that a person sits on it; the purpose of a bed is that a person lies on it (and probably sleeps on it); etc. In our model, when the utility (purpose) of a piece of furniture is affected, the system triggers a Tension of Utility. In this way, if a chair is situated on the bed, which implies that neither the bed nor the chair can serve its purpose anymore, the system detects this situation and triggers two Tu (one for each piece of furniture); if the chair is removed, both tensions are deactivated.

Tension of Proximity (Tp). In our model, the distance between each individual piece of furniture must be at least 10 centimetres (with the exception of the chairs located by the dining-room table). Thus, if the user locates a chair in front of a coffee-table side by side, a Tension of Proximity is activated. In other words, when two or more pieces of furniture are too close to each other (less than the equivalent to 10 centimetres) the system triggers a Tp.

Tension of Distribution (Td). In our model, it is important that the furniture is evenly distributed around the room. For example, if a bed is situated on e2, a chair on c3 and a table on e3, only the right part of the room is occupied. All the left part of the room has no furniture and therefore looks strange. So, if the system detects this situation, it triggers a Tension of distribution. We decided that a piece of furniture situated at c does not trigger a Td.

In the current version of the system, all tensions have a numerical value established by the user. Thus, we can calculate the value of the tension produced by a specific arrangement of furniture in the room. Figure 3 shows three possible arrangements of household goods. Figure 3.b shows a *vase* on the floor at c, a *bed* on the floor with its head pointing towards the north at c_3 (it invades e2 and e3 due to its size and orientation), a *chair* on the bed at c_3 and armchair at c1 pointing towards the west. This arrangement triggers the following tensions: two Tensions of Utility (Tu) due to the chair on the bed (the bed cannot be employ to sleep and the chair cannot be employ to sit; each tension has a value of 4); a Tension of Distribution (Td) because positions e1, c2, e4 and c4 are empty while the rest contains (at least part of) a piece of furniture (this tension has a value of 4). This arrangement generates a tension = 12.Thus, it is possible to describe a specific room-design in terms of its furniture, their position in the room and the tensions they produce. We refer to this description as *Context*. So, the Context of figure 3.b is comprised by: a *vase* on the floor at c, a *bed* on the floor at c_3, a *chair* on the bed at c_3, an *armchair* on the floor at c1, two Tu due to the chair on the bed and one Td due to four empty positions. *Context is an important part of our model since it is employed during the generation phase (engagement) as cue to probe memory and retrieve a possible action to continue the design of the room*.

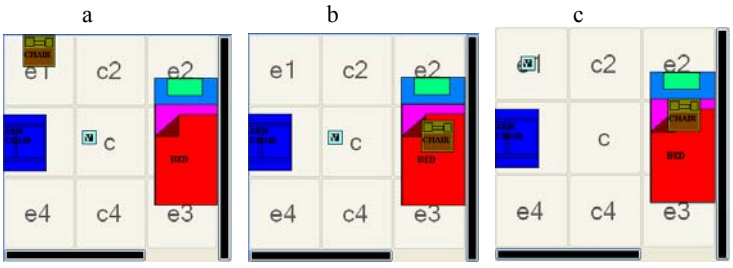


Fig. 3 Three possible arrangements of furniture within a room.

In our current model, it is possible to generate two types of designs: calm and tense. A calm-design has a Tension < 10, a tense-design has a Tension > 10. We associate a calm-design with a conventional disposition of household goods within a room; we associate a tense-design with an unorthodox disposition of household goods which attempts to produce an emotional reaction in those individuals that watch into the room.
The design process implies to generate a sequence of different arrangements of furniture until the right one is found. For example, figure 3 shows a sequence of three different household goods dispositions. The Context of Figure 3.a, named as Context1, is comprised by: a vase located at c, a bed located at c3, a chair located at e1, and an armchair located at c1; a Td is generated because positions c2, e4 and c4 are empty. In figure 3.b an action has been performed to modify this design: the chair has been located on the bed. This action produces a new context (described some lines above) named as Context 2. In figure 3.c a new action has been performed: the vase is moved to e1. This action produces a new context in this example named as Context 3. The final design has a tension = 12 so, it is classified as a tense-design.
As an important characteristic of our model, we link contexts with possible actions to perform; e.g. Context1 is linked to the action "put the chair on the bed"; Context2 is linked to the action "Put the vase on corner e1" In this way, we are able to generate sequences of actions during design by associating different contexts (see Section 4). However, because contexts describe very concrete situations and the E-R model requires more general representations, we transform contexts into what we refer to as *Atoms*. An atom is a context where all concrete objects (e.g. table, bed) and their positions in the room (e.g. e1, c3) are substituted by variables. In this way, Context1 and its linked action are transformed into Atom1:

Atom1: Object1 is located at cX, Object2 is located at cY, Object3 is located at eX and Object 4 is located at eY; Td= 3 (because three quadrants empty).
Linked Action: Object3 is put on Object 2.

Object1, Object2, Object3 and Object4 are variables that can be instantiated with any furniture; cX and cY are variables that can be instantiated with any center; eX and eY are variables that can be instantiated with any corner. The linked action indicates that Object3 must be put on Object2. Notice that the relations between objects and locations are kept and each variable represents a different element. *Thus, the goal of this project is to associate Atoms (sets of tensions and furniture-locations relations) with possible actions to perform. Then, employ this information to generate during engagement novel arrangements of furniture*.

4. General description of the system

As mentioned earlier, our current prototype only implements the engagement part of the E-R model. The system has two main processes: building the knowledge structures (atoms) and the design process (engagement). Engagement employs atoms to generate novel designs.

Building knowledge structures. In order to generate atoms we have developed a computer application to perform a design process (user-mode). This application shows a representation of the room and the furniture that we described earlier. The user can perform any of the following actions: put a piece of furniture on any of the defined positions; eliminate a piece of furniture from the room; move a piece of furniture from one position to a new position; put a piece of furniture on top of another piece of furniture; change the orientation of a piece of furniture. The aim of this program is to permit the user to generate different designs. The program records user's actions and generated contexts, and employ these information to build its atoms.
Engagement (design-mode). The user can ask the system to generate a tense or a calm design. In a tense design the system employs actions that increase the tension; therefore, actions that decrement the tension are considered as not useful. In a calm design, the system employs actions that increase the tension and then actions that decrease the tension. During the decreasing phase, actions that increment the tension are considered as not useful. Engagement works as follows:
1. An initial arrangement of furniture (a seed) is provided to the system by the user.
2. The system calculates the positions of the furniture and the tensions they produce in order to update the Context.
3. The context is employed as cue to probe memory. The goal is to match all those atoms that are at least 50% similar to the context (this parameter is known as the ACAS-Constant and can

be modified by the user. Its purpose is to give the system the possibility of matching atoms that are not identical to the Context. In this way, novel designs emerge. See Pérez y Pérez 2007). The system retrieves all actions linked to the matched atoms.
4. Those actions that are not useful for the design in progress are eliminated. From the remaining actions one is selected at random as the following action to continue the design.
5. The system goes back to step 2.
The cycle ends when an impasse is declared (no atom can be matched), when a predefined number of actions have been performed or when a predefined value of tension is reached

5. Experiments and Discussion.

To test the model we provided the system (employing the user-mode) with eight different tense-designs and calm-designs. Then, we asked the program to generate designs which were original (different to those that we provided) and which were either tense or calm. The system was able to accomplish the task. Figure 5 shows an example of the steps that the program followed to develop an original tensional design.

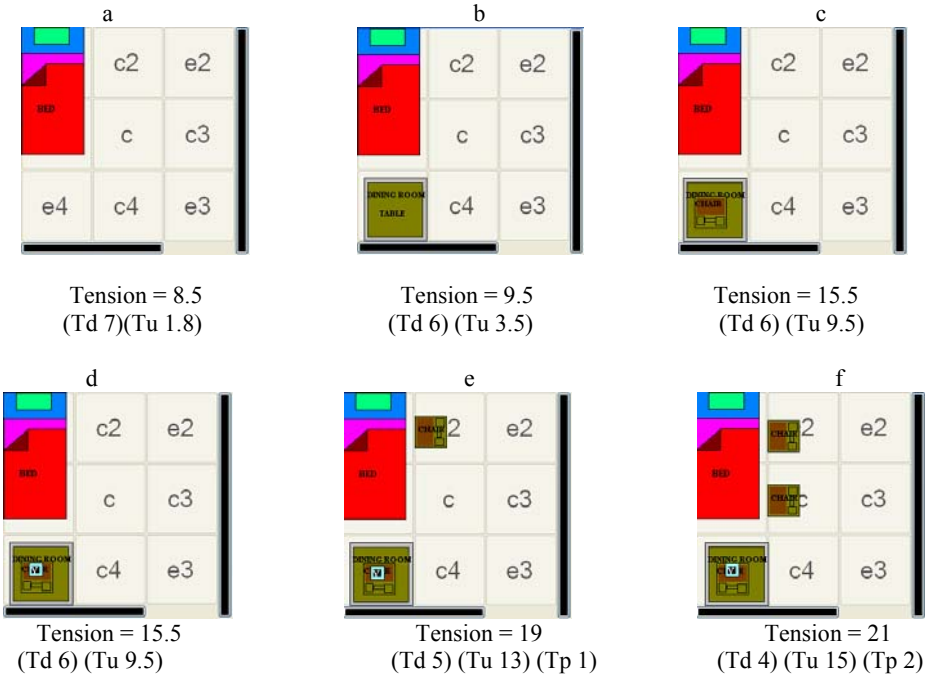


Figure 5. Steps to produce a tensional design.

The bed on the upper-left corner was provided by us as the initial context (see figure 5.a). Then, based on the experience recorded in its knowledge-base, the system surrounded the bed with furniture that increased the tension (a table with a chair and a vase on top of it, and two chairs located too close to the bed). That is, the system locates the furniture in the room based on its previous experience (not at random) and is able to satisfy the requirement of generating a novel arrangement which produces tension. In other experiments, when we set the ACAS-Constant to a high value (e.g. 90%) the system starts copying previous designs (instead of creating novel ones) and impasses are easily declared (i.e. the system cannot match any atom). This is congruent with previous results [7].
This paper reports a work in progress. Based on the E-R model, we have developed the engagement part of a program that generates novel arrangements of furniture in a room. To achieve this goal, we defined how to represent emotional reactions to furniture arrangements, and how to build Contexts and Atoms (set of tensions associated to actions). In this way we have shown that: 1) the engagement part of the E-R can be employed to generate original

dispositions of household goods in a restricted domain; 2) representations of tensions can be employed to guide the design process.

These are just initial results. We expect that once we implement the reflection part, where the material produced is evaluated for novelty and interestingness, and if it is necessary the material is modified by the system, our results will improve. That is, we will be able to generate more interesting designs. In the same way, we plan to extend our work by including more types of tensions, more furniture, as well as adding features like illumination and colour to the room and household goods. We believe those elements play an important role in the appraisal of emotional reactions.

We hope this work encourages people to study the role of emotions in the creative process.

> In the 1980s, in writing The Design of Everyday Things, I didn't take emotions into account. I addressed utility and usability, function and form, all in a logical, dispassionate way—even though I am infuriated by poorly designed objects. But now I've changed. Why? In part because of new scientific advances in our understanding of the brain and of how emotion and cognition are thoroughly intertwined. We scientists now understand how important emotion is to everyday life, how valuable. Sure, utility and usability are important, but without fun and pleasure, joy and excitement, and yes, anxiety and anger, fear and rage, our lives would be incomplete. [9]

References

[1] Pullman, M. E. & Gross, M. A. (2004) Ability of Experience Design Elements to Elicit Emotions and Loyalty Behaviors. *Decision Sciences*, 35 (3), 551-578.

[2] Ritterfeld, U. (2002) Social Heuristics In Interior Design Preferences. *Journal of Environmental Psychology*, 22, 369-386

[3] Mehrabian, A. & Diamond S. (1971). Seating Arrangement and Conversation. *Sociometry, 34 (2), 281-289.*

[4] Lang, D. C. (2002). *Teacher interaction within the Physical Environment. Ph.D. Dissertation, University of Washington.*

[5] Pérez y Pérez & Sharples 2001. MEXICA: a computer model of a cognitive account of creative writing. *Journal of Experimental and Theoretical Artificial Intelligence, 13(2), 119-139*.

[6] Gelernter, D. 1994. The Muse in the Machine. London: Fourth Estate.

[7] Pérez y Pérez, R. 2007. Employing Emotions to Drive Plot Generation in a Computer Based Storyteller. *Cognitive Systems Research. 8(2), 89-109.*

[8] Acosta Villasenor, E. & Pérez y Pérez, R. (2005). The Geometrician: a computer model for problem solving in the field of geometry. In P. Gervás, A. Pease and T. Veale (eds.), *Proceedings of the Second Joint Workshop on Computational Creativity, at the 19TH International Joint Conference on Artificial Intelligence* (IJCAI'05), (pp. 10-16). Edinburgh, Scotland.

[9] Norman, D. A. (2004). *Emotional Design: Why We Love (or Hate) Everyday Things*. New York: Basic Books.

Author Index