Computational Poetry Workshop

I AM THAT I AM

Abstract

Creativity cannot exist in a vacuum; it develops through feedback, learning and social interaction with others. However, this perspective has been relatively under-investigated in computational creativity research, which typically examines systems that operate individually. We develop a set of requirements for creativity systems that can incorporate communication and give and receive feedback. FloWr is intended as a paradigm in which computational creativity software can be implemented and integrated. We ask if the FloWr framework can model social creativity, and if not, what changes may need to be made in order to accommodate the requirements of social creativity. As a thought experiment to test these ideas, we consider a case study of how feedback and dialogues can help develop the creativity of computational poetry writing, and how such a system could be implemented in FloWr. We also consider alternatives to FloWr for modelling social creativity.

Keywords: poetry, social creativity, flowcharts, distributed computing, population-based methods, agent-based models, duration, temporality, serendipity, design patterns, Writer's Workshops

'We *can* talk,' said the Tiger-lily: 'when there's anybody worth talking to.'

Through the Looking Glass, Lewis Carroll

Introduction

We should clarify at the outset that a central part of this paper is not new. We are writing this paper in a large part to champion Alan Turing's proposal that intelligent machines should "be able to converse with each other to sharpen their wits" (**turing-intelligent**). What is new is the particular approach to computation that we advance in support of this proposal.

While we do not precisely suggest an alternative to the input/output/process model of computation (**marr1981vision**), we do propose to extend it to build social effects. A lot hinges on the understanding of the term "social." The formalism that is proposed here builds on the notion of second order cybernetics, as encapsulated by these two propositions of Heinz von Foerster's:

"Anything said is said *by* an observer."
"Anything said is said *to* an observer."

(**von2003cybernetics**)

Von Foerster characterises *first-order cybernetics* as "the cybernetics of observed systems" and *second-order cybernetics* as "the cybernetics of observing systems." Together with the propositions above, this points to a connection between observers, language, and a relationship in which the language is used. We call the indicated relationship "social." While the typical genre of design and programming is concerned with specifying a system's purpose (and its inner workings), the aim in second-order cybernetics is for a system participant to specify its own purpose in a "social" manner, that is, in relationship to the rest of the system.

This is closely related to W. Brian Arthur's way of thinking about the science of complexity. He suggests that for sufficiently complex problems, there is no correct statement of the problem. "All you can say is that you have this situation and there are many ways to cognize it" (**brian-arthur-interview**). The consequence is that:

> Complexity is looking at interacting elements and asking how they form patterns and how the patterns unfold. It's important to point out that the patterns may never be finished. They're open-ended. (**brian-arthur-interview**)

The aim in this paper is to enhance the computer's response-ability, its ability to identify its own approach to a given situation, including its ability to identify and encode new patterns. We can potentially apply this sort thinking at each level of a program. In this way, the model input/output/process morphs to become something more like *context/response*. A response may be in the form of "output" or it may be in the form of "process" or it may be a mixture of these. In particular, a satisfactory response may be for the entity in question to change its own structure. Another part of the response might change the context. This could all be viewed in a formal way as "output" – new code may be written, to update the entity's definition of "self," for example – but this isn't quite the same as the standard linear model that deals with components that have been precisely defined in terms of its input and output. We still have components, but their behaviour can change over time.

An approach like this will be necessary for building autonomous (and even adaptive) machines. Equally, this approach seems necessary for the development of a theory of *social cybernetics* as outlined by Von Foerster. In the context of literature rather than programming, similar comments had been made over a decade earlier, by Mikhail Bakhtin, who wrote:

> The thinking human consciousness and the dialogic sphere in which this consciousness exists, in all its depth and specificity, cannot be reached through a monologic artistic approach.
>
> (**bakhtin1984problems**)

Predicated on the assumption that *computers are not human* (**colton2012painting**), computers *must* be social, if they are to deal with problems of any great complexity (**minsky1967programming**; **society-of-mind**).

This paper outlines an approach to social creativity, taking the domain of computer poetry as our working domain. It uses the Writer's Workshop model (**gabriel2002writer**) as its primary thought-architecture. It will be of interest to others working in the field of computational creativity (hereafter, CC), who, we hope, will read it as an invitation to a dialogue.

Background: Social creativity in CC

Although we have adopted the term "social creativity" following (**saunders12**) and with the specific understanding developed above, we could also refer in a similar spirit to situated, interactive, communal, contextual, conversational, group, dialogical, discourse-based, community-based, interaction-based, or feedback-informed creativity.

The point is that creativity cannot exist in a vacuum. The very essence of creativity lives in its appreciation by the creative entity themselves and its audience. As we have remarked elsewhere, creativity is in the eye of the beholder. During the creative process, self-evaluation abilities are crucial (**poincare29**; **csik88**). Social creativity expands upon this paradigm by bringing co-creators into the process, and creating works that rely on dialogue, reflection, and multiple perspectives. The "results" may be steeped in process and will not always be based in consensus.

The Four Ps of creativity – the creative Person, Product, Process and Press (i.e. environment) (**rhodes61**; **mackinnon70**) – have been emphasised in general creativity research. *Pluralising* these terms would call attention to a social dimension of creativity, and leads to a more inclusive and encompassing approach to the study of creativity – one that accommodates multiple perspectives. The Pluralised Ps remind us that it is not sufficient to model a lone creator or to generate an attractive artefact. To model creativity more completely, we also need to consider the environment in which a creative person operates, and how the environment is used in the creative process.

Computational creativity research has achieved many successes in computational generation of creative products. The question of how these systems could adapt and learn from feedback to improve their creativity, however, remains underexplored in computational creativity despite evaluation being a pivotal contributory part of the creative process. Researchers have generally preferred to take on the task of generating artefacts that could be seen as creative, as a neces-

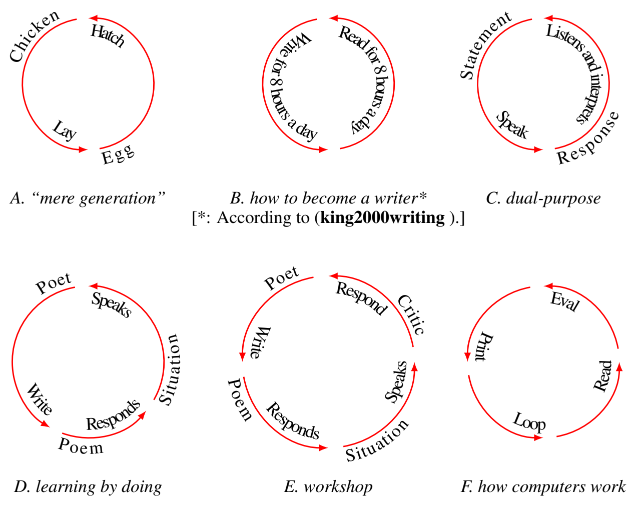

Figure 1: Several illustrative cycles

sary prior to the task of incorporating self-evaluation within a creative system (**jordanous11iccc**).

Some notable exceptions exist, highlighting the importance of the environment in which a creative system is situated (**mcgraw93**; **sosa09**; **perezyperez10MM**; **pease10**; **saunders12**), with some of this work influenced by the DIFI (Domain-Individual-Field-Interaction) framework (**csik88**). Generally, however, social interaction between creative agents and their audience is an area which has been neglected. Increased development of the interactivity of creative systems, especially where this affects the way these systems works, is pleasing to see and deserves further attention (**coltonwiggins12**).

Other CC material? Anything about poetry in particular? It might be a good idea to mention Tristan Tzara, Brion Gysin, etc. Probably at least a few words about FloWr here as well.

E.g. (**jordanous10**)? Definitely we should cite (**misztal2014poetry**) and explain a bit about the MASTER system.

Philosophy and methods

Social and Aesthetic Philosophy - TBA

Key ideas: social, language, emergence, dialogue, training. . . what else?

Relatively mainstream philosophy: Hume, Bergson, Mead, Dewey, Bakhtin, Gilligan, Nietzche, Sloterdijk. Aesthetic philosophers, and poets writing on aesthetics: Collingwood, Colleridge, Carver, Keats, Kant.

Methods: "What are the proposed 'lab rats'?"

There are many possible places for a "dialogical" intervention within the writing process; see Figure 1. Figure 1(A) shows the standard chicken-and-egg problem, which provokes the question of *evolution*; 1(B) shows an analogous picture that gives a simple recipe for the growth and development of a writer; 1(C) is a formally similar diagram that squares up to the metalinguistic features of the situation, showing that a *response* (which may be verbal, visceral, physical or something else) always has dimensions that goes beyond the utterance that is overheard; 1(D) examines in further detail what happens when someone *writes* – namely, writing as a response to a situation that may allow the writer to make sense of this situation; 1(E) adds a *critic* who responds to the situation, particularly as it is expressed through and enhanced by the poem; 1(F) shows that this situation is not an entirely unfamiliar for computer programmers, especially considering that the "Eval" phase in a Read-Eval-Print loop be interrupted with a debugger to fine-tune program operation.

In (**serendipity-arxiv**), we described a related template for a pattern language for interactions in a computational poetry workshop:

> `presentation`, `listening`, `feedback`, `questions`, and `reflections`.

To this should be added the potential for real-time `replies` to `questions` before subsequent "offline" `reflections`. We used this template to expand several of the patterns of serendipity described by (**van1994anatomy**), showing how they could be used to foster discovery and invention in a workshop environment.

Our "lab rats" are, accordingly, not poems – which could, after all, be developed through "mere generation" – but are, rather, *instances of reading and responding to poetry*. This may take place within a formal "workshop" context or they may take place in smaller-scale experiments where a computer system reads and responds to other poets. Naturally, such responses can also be more or less "canned" (as with Michael Cook's humorously nonspecific AppreciationBot), so the question becomes what constitutes an authentic, interesting, or useful response, and how will these be developed? The idea of responses is also useful at the micro-level, as will be made clear in the following section.

Design

Questions to ask when reading poems

Table 1 contains a list of questions that could be addressed, in a programmatic manner, to analyse an individual poem. These questions, coming from a computer poetry perspective, do not make the same assumptions about linguistic understanding or embodied experience that apply to human poets. It is worthwhile to compare this list with a list of questions that a reasonably sophisticated poetry reader might ask about poems (Table 2). The difference between these two frames of reference is most instructive.

In the approach to "sophisticated" reading considered here, we focus on the process-oriented question: "What does the poem tell us about how it was made?" This can be studied through the lens of the follow-up question "What are the aspects of the poem and how are they present to the reader?"

Question	**Agent concerned**
Word level	
What does this word mean?	WordNet expert
Where does this word come from?	Provenance expert
Phrase level	
What is this line about?	Keywords expert
Line level	
What is the sentiment of this line?	Sentiment expert
Is this alliteration, rhyme, consonance, etc. important?	Style expert
Poem level	
How many rhymes are in the poem?	Rhymes expert
How well does the poem's metrical structure flow?	Rhythm expert
How repetitive is the poem?	Repetition expert

Table 1: Questions we could implement using various computational agents

I've compressed this down a lot. Add some references and expand this a bit. Our old notes may be useful.

Among the several aspects that may be present in the poem are register(s); addressee(s); position(s); the changing awareness of the reader; character(s); image(s); functions, mechanics, and paradigms; problems, discomforts, and dis-easements; apparent versus substantiated truths; overlapping scenarios; chronology of reading and chronological references; lexical choices that carry a deeper semantic meaning; allusive effects; and manifest uncertainty in the poem's construction.

One of the striking things about this list is that it does not easily divide itself into "levels" in the same way as the items in Table 1 do. For instance, even though lexical effects clearly have to do with an analysis taking place at the "word level" the task that a given selection of words performs is typically global; that is, it is meaningful at the "poem level." The questions in Table 2 may themselves be divided into registers and positions. The process of reading a poem is also a process of *poiesis*. Each of the examples listed in the right-hand column of this table (and a plethora that are not listed) plays a role in the "society of mind" of a reader, analogous to the agents in Table 1.

Bridges between 'theory' and 'practice'

There are things we can actually point to in poems, and matching concepts in aesthetic philosophy. Furthermore, we can actually approach this scientifically. The computational approach to poetry is one way to build the practice/theory bridge. In particular, our *ansatz* is that the workshop could serve as a way to deepen an understanding of a poem's semantics!

Question	**Examples**
What is are the register(s) of the poem?	cliched, instructive, imperative
Who is addressed?	friend, rival, lover, confidante, pupil
What position(s) are present in the poem?	pleading, remonstrating, ephemeral
What becomes of the reader whilst reading the poem?	alienated, perplexed, amused
Who are the characters in the poem?	"the falconer", "you", narrator, "two men"
What is the role of image(s) in the poem?	"the sea", "a bicycle"; multiple meanings
What functions, mechanics, and paradigms are present for the reader to engage with?	communication, subverted cliche
What problems, discomforts, or diseasements are invoked in the poem?	terror, self-loathing, rejection, desire
How do these evolve?	E.g. an image may start to take over from a register
What *is* in the world of the poem as compared with what you only think is?	"Surely", "must"; sacred vs mundane; perspectival vs surreal
What are the overlaps, transitions, implicit dialogues?	"twinned" lines/ideas, juxtaposed parts of the poem
What role does the chronology of reading play, versus references to chronology and chronological positions within the poem?	flagged development, evolution, movement, stasis
How are lexical categories used?	flimsy adverbs, solid nouns, thudding adjectives
Are there discernible allusive effects?	illustrating the literary apprenticeship of the author (or reader)
How does Keats' idea of negative capability feature in the composition – "that is when man is capable of being in uncertainties"?	we must worry about overconfidence, overdetermined lines

Table 2: Questions that we actually ask when reading a poem

There are certain prerequisites. In addition to the presentation of written work, an underlying situation is assumed, one that is shared (with respect to differing points of view) by the poet and the critic (see Figure 1). The poet and critic are assumed to have relatively stable, enduring but evolving, identities – so that it would be possible at a given juncture for either one to consider the question "Who am *I*?" and "Who are *you*?" (**bakhtin1984problems**).

To begin with, in response to a given computer-generated poem:

Oh dog the mysterious demon
Why do you feel startle of attention?
Oh demon the lonely encounter
ghostly elusive ruler
Oh encounter the horrible glimpse
helpless introspective consciousness

A human critic might offer the following feedback:

1. The use of the word *mysterious* in the first line has no resolution, real or attempted, or quest to find one.
2. The use of the word *attention* is not being interrogated or acknowledged for its importance. Its qualifying word is *startle*, used here as an adjective; acknowledging the fact that the attention is noted but not yet part of the transformative of the poem.
3. This is repeated in the next references to the aesthetic experience as a *lonely encounter* and an *exclusive ruler*, which is then qualified as *a horrible glimpse*.
4. So reference to the contact made between the poem and its own event are made through the words *demon*, *encounter* and *consciousness* and all of these are qualified in negative terms.
5. This poem does not welcome the intimacy of bringing anything to aesthetic consciousness so that it might be expressed. Why do I say that? Because the words are generalised and *horribly* imprecise.
6. The really interesting thing about it is its own apparent understanding that this is so. Look at the words *mysterious*, *feel lonely*, *elusive*, *horrible*, *glimpse*, *helpless*, *introspective*, *conscious*. They are unspecific – nothing has been explored as the poem moves toward a better understanding of these ideas. They describe but they do not illuminate by becoming anything else. They all associate exploration with fear and isolation and this is (paradoxically) quite an interesting acknowledgment of the poem's refusal to go anywhere i.e become a thing transformed by a creative process.

Each of these comments is *dual-voiced* in the sense that the critic is relaying the poet's speech with a new emphasis. Each such statement is one side of a micro-dialogue (**bakhtin1984problems**). The challenge is, of course, to bring the observations into the awareness of the poet, across the "silicon divide." From the programmer's standpoint, this involves massaging each of the observations into a language that the computer can understand – the most obvious candidate being the programming language the computer used

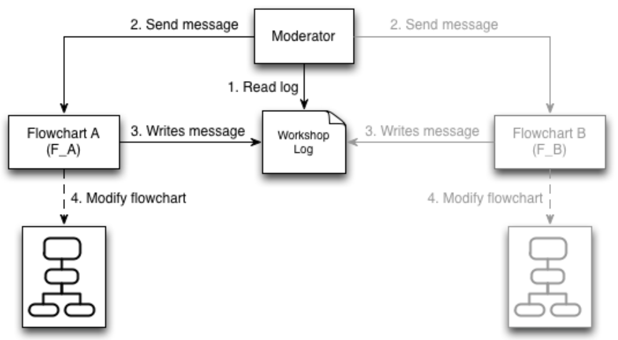

Figure 2: Schematic design for a workshop built in the FloWr system

to generate the poem. But care should be taken not to just blythely program the computer with more rules, but rather to give attention to the process of learning new rules contextually.

Rather more briefly, let us consider a reversal of roles, and put the computer in the position of critic, looking at a passage from an historical piece of poetry. We have selected one that might have – but in fact did not – serve as a model for the poem generated above.

I'm truly sorry man's dominion
Has broken Nature's social union,
An' justifies that ill opinion
Which makes thee startle
At me, thy poor, earth born companion
An' fellow mortal!

Naturally, the first problem is for the computer to *read* the poem. There are various possible approaches to this problem. One of the approaches that is most appealing from our point of view is the automatic generation of a semantic network from the input text (**harrington2007asknet**). This, again, could be enhanced by additional meanings that are not just factual, say, but aesthetic (drawing on the notions informing Table 2).

We do not propose that either of these programming tasks is entirely easy, but both are entirely feasible.

Prototyping a workshop in FloWr

Development plan (Christian):

- Define workshop, including product properties, e.g. variety, innovation, etc. Alternatively, refer to prior definition earlier in the paper and point out benefits again.
- Introduce different roles in a workshop: master, working student, criticising student (anything else?). These roles can be combined into more complex ones.
- Introduce tasks they perform: introducing and demonstrating new tools, using tools to generate, analyse to improve creation, etc.
- Show the demon dog flowchart as part of the FloWr system, and highlight which roles the different parts represent, and which tasks are present.
  - A flowchart could be identified as a student that both creates and asseses (fitness functions) at the same time.
  - The script/moderator could be identified as the master who guides the student, tells it what to do in which way.
  - The set of all process nodes can be understood as a toolbox the master can access, to hand tools to the student.
- Show what's missing to fulfill the prior definition of a workshop
  - Multiple students that collaborate and help improving each other's creations.
  - Master as mediator between two students, asking one student what tools it uses, and telling the other to adapt them.
  - The moderator can introduce new tools to the workshop.
  - Students must be able to notice at one step of the creation process (e.g. while using one tool), that the usage of a prior tool could be improved. They could then either first finish or rewind.
  - Allow for more than one agent. Allow for more than one agent to work at a time.
- Show which features are required to realize that:
  - Understanding the workshop as a hierachical multi-agent system, where the master is at the top and the students are at the bottom of the hierarchy.
  - Simplification by separating flowcharts that create and assess from those which only assess.
  - Thus introducing external assesment.
  - Thus communication between flowcharts required.
  - Enable an agent (more general, e.g. a student) to explain what tools it is using, in which order, and which parts of the output they affect.
  - Allow other agents to improve their flowchart by means of this information, e.g. add a particular tool in a particular step.
  - Enable master as mediator.
  - Allow master to map observations of poems into new tools, that are made available to the workshop.
  - Allow communication between process nodes, i.e. tools: E.g. assessment in rhyming node shows that too few words/sentences available for good results. Thus necessary to fetch more/different sentences in a prior step. Problem: agency when using the term "tool" - separates assessment by the agent from using the tool for creation.
  - Introduce parallelisation, to allow multiple agents to work at the same time.
- Open questions:
  - Decision making: It's probably more important to answer how decisions are made to adapt another tool, etc., than stating where they're made. E.g. decision making by means of a fitness function of the current artefact, by means of curiosity, etc.
  - Add more implementation suggestions

– Would it be better to focus on one new aspect of the workshop, and leave the others out? E.g. add master who can introduce new tools, or introduce multiple students that can communicate. One paper might be not enough to get from current FloWr to a full-fledged workshop.

What's the development plan look like? Should we show the flowchart for the "demon dog" poem as a very early prototype and concrete example to critique? We could illustrate – in principle – how it would be improved in a workshop.

We'll need to discuss the different levels of the design. E.g.

Moderator.

Flowchart A: "I do this" (service advertising what facilities it has available: if we can swap them in and out, we could do some kind of A/B test to check whether swapping in a different node actually improves the result according to some critic or other).

Flowchart B: "I would like to be able to do that too." Temporality. More feedback from critical agents. Learn and adapt the script.

Where is the decision made to modify (e.g. only by the Moderator, or at the level of individual flowcharts, or nodes, etc.?)

This is relevant to the issue of "parallel solutions". At the level of some downstream process: How do we decide which of the parallel solutions to pick and carry ahead to the next step? Do we give feedback to the previous step in the process?

If we are allowed to take multiple different paths, not just parallel copies of the same process, where is the decision made as to which path to take?

Central control Distributed control Autonomous Global wiring

Critic: "This 'Twitter' node is not good."

Follow the idea of the prepared mind, add any problem that is noticed to a list of "problems to solve." Again, the log of suggestions/outstanding problems could exist at different levels/timescales. (NB. And a similar approach can apply for "questions" as well as "problems." This "log" doesn't just have to be a list, it could also be a frame or flowchart etc.)

Protocol:

flowchart to flowchart?

Do we also need some modification for node-to-node communication? Alternatively, are we going to follow Christian's suggestion and merge flowcharts and nodes into one kind of entity?

Also, do we need an overall "protocol" for the Workshop demo application, different from the more general communication protocol that is used?

There will be three different type of messages: questions, answers and suggestions.

Questions can be about sources of information; e.g. files, online articles, input from another node, etc., about elements of poems; e.g. similes, rhymes, keywords, sentiment, etc.; about specific details; e.g. count, purpose, etc. Answers would be associated to previous questions, and suggestions are changes proposed by one system to the other.

Commentary is needed in order to enable the dialogue between flowcharts. Each node in FloWr has two main components, a set of input parameters and a set of output variables. Parameters can either be sources of information; e.g. the Guardian newspaper, or conditions; e.g. range of dates. Variables are outputs of different types. A commentary would be added to each of them in order to facilitate communication between flowcharts and nodes. This will enable the dialogue between the participants in the workshop; i.e. ask and answer questions, as well as it will allow to identify where to apply suggestions for new versions of a flowchart as suggested in a workshop session. The proposed commentary is presented in an Appendix.

Do we consider e.g. genetic algorithms present (i.e. generate) several options and allow the downstream nodes to comment on each one high level feedback (on the whole generated poem) and micro-feedback (at the level of individual nodes)

Atomic nodes vs composite nodes (flowcharts). Can we just forget about what level we are at? Node or flowchart – and use a recursive approach that can be applied to atomic or composite nodes? (Christian's diagram of the two different kinds of communication – between flowcharts or between nodes – reduce to one kind of communication if we take the recursion approach. Similarly, reification then becomes possible. In itself this isn't a huge technical advance, but it might allow us to deal with examples of "morphogenesis", which is interesting. E.g. grow until I have 100 process nodes and then stop. That said, "reification" is a bit frowned upon – but provenance is always a good thing.)

Discussion

Potential applications

Eventually we would like to see the paradigm advanced here in effect across CC. This doesn't mean that we would remove the "generation" aspects of CC, but that we would pair them more closely with reflection. The workshop method of critique may shift to more closely model an *atelier* method of creation. The same skills that support learning in a writers workshop may support a form of dialogue with the work itself, leading to richer creative artefacts that show us more about the creative process.

The workshop brings a range of practical and philosophical challenges for CC, and the broader field of AI, related to asking and answering novel questions, the practicalities of learning over time, a sense of identity and personal style.

Focusing on these questions does not in any way suggest that we should devalue works from lone creatives, but it does suggest that we think about how we knit individuals together in the social fabric of the CC community. The current model at the International Conference for Computational Creativity (ICCC) is similar to many other academic conferences, even though our subject matter is really quite different. It's all well and good for us to travel and present *our* work to one another and build *our* sense of community in that way, but what about a track for computers to present their work?

As (**turing-intelligent**) foresaw, computers have gotten quite good at Chess and reasonably good at Go, using methods very similar to the workshop. Should we not follow their lead? Poetry seems a natural next step; prose literature may be approachable through similar methods. Indeed, why bother writing dialogue for a NaNoGenMo[1] novel, if you could simulate it?

Potential criticisms

One class of criticisms could relate to the appropriateness of dialogue per se: "Why not put everything into one flow chart? Or one node, for that matter?" In cases where dialogue is indeed seen to be necessary, a different sort of question arises, namely, how do we know if we're doing it well? E.g. how will we avoid the pitfalls of "design by committee"?

We believe we have adequately addressed the first set of questions, following Bakhtin. The second set of questions will have to be worked out in practice, but we should keep in mind the potentially greater pitfalls of asocial design. Indeed, we wonder if apparent failures of social creativity are often due to a poor grasp of the social rather than an overly social approach.

Another line of questioning that we may expect from some CC practitioners is as follows. Given the historical emphasis on *creating* new artefacts in CC, shifting the emphasis to the computational *appreciation* of already-created artefacts is somewhat strange. More pointedly, members of the public may say: computers creating art is bad enough! Surely, you don't expect them to *study* too?

We have portrayed this criticism in somewhat facetious terms. The economic questions posed to CC by its critics are real, but building programs that are not able to stand up for themselves and that are out of touch with historical and modern trends in art (or other fields of human endeavour) is hardly the answer.

The actual problem is that appreciation of computationally created artefacts is *hard*. Consider the difference between creating a video game (for example) and playing a video game. In the first case, the designer has full control over the rule-set, game mechanics, interaction devices and so forth. In the second case, we're more or less in the world of general AI. It is of course less untoward for a computational video game designer to play its own games; this is state of the art.

While the sketch of a solution developed here is by no means complete even for poetry, we believe that extending capabilities from both sides is robust. However, as none of this has been tested in detailed experiment, a fair criticism is that we do not know, yet, either how much more work this approach will be, or how much better the results will be. We suspect it will be both harder, and worth it. In the following section we discuss the future outlook for the research approach.

[1]https://github.com/dariusk/NaNoGenMo

Conclusions

We have made the case for dialogicity in CC and in CC research, building on a comparison of different ways to read poems, and considering the potential of an existing software system to support dialogue at various levels.

Our requirements for communication points to the need for program elements to be able to learn and adapt. Genetic programming (**koza1992genetic**) and related methods offer a certain precedent for this way of working.

The current paper has focused on more global concerns. The current FloWr "ecosystem" focuses on features like user interface ease for human programmers, plus reusability of modules. The next steps should involve an agent-based redesign. We pointed to a considerable amount of related work in an earlier section. Off-the-shelf open source software exists to support some relevant basic features.

However, the question of *context* has not been adequately addressed either in CC or in mainstream programming. While we do not agree with interpretations of artwork that macro-reductionistic in the sense that "the environment determines the artwork" nevertheless there are important contextual effects (**geertz1976art**), and artwork generally involves an engagement with context. Philosophically this situates the work in the realms of *pragmatics* (**sep-pragmatics**) and *metalinguistics* (**gombert1994development**). Practically speaking, the questions relate to building sophistication in the specific "metalanguage" of programming. The programming tasks involved are not necessarily ends in themselves, however, but oriented, in the first place towards building better artefacts. These too have a goal, which (at least in a simple case) is to communicate.

The graphical programming environment that shipped with the 1984 robot simulation game ChipWits, not taken seriously as a competitor to FloWr, does nevertheless have some useful programming facilities, like the ability to loop and run different paths in the flowchart depending on conditions, and the ability for one flowchart to reference another one, as with spreadsheets. However, it does not have the ability to self-program, which is the essence of the proposal for FloWr. The ability to generate-and-check (using a population-based mechanism) and more importantly, the ability to learn over time (as we explored from a formal perspective in (**colton-assessingprogress**)) are two other features that should come standard in future versions of FloWr. We want to be able to take account of the abundance of available information, to introduce "noise", and fully take account of the existing "framing" in relationship to the context/situation.

Although social creativity has been explored in CC, it tends to be an exception rather than the rule. In future work, we'd like to be able to say "We've done it!" although to develop a concrete proof of concept we will have to focus on a few simple metrics (like musicality).

There is also a “social engineering” component to this paper, hoping to motivate a new approach to research evaluation in CC. Between the kind of design research carried out in this paper and a large-scale double-blind study that would “prove” the superiority of a social approach is something more contextual, involving the actual practices and abilities of participants (**seikkula2006dialogical**). We sketched an approach to the evaluation of poems, but similar thinking can help develop an evaluation of programs. The question of how much architecture should be shared in the CC community: do we need a shared “CCyc” platform, or should we “let a hundred flowers bloom”? One moderate approach would be to revive the *floral games* of the troubadours.

Acknowledgements

This research has been supported by EPSRC grants EP/L00206X and EP/J004049, and the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open Grant numbers: 611553 (COINVENT) and 611560 (WHIM).

Appendix

Proposed message framework for use in question-answering

```
<message>
id: s1_m1
type: question
source_system: F_A
target_system: F_B
what: source
element: theme
</message>
```

```
<message>
id: s1_m2
type: answer
question: s1_m1
source_system: F_B
target_system: F_A
answer: guardian article
</message>

<message>
id: s1\_m3
type: suggestion
source_system: F_A
target_system: F_B
what: source
element: theme
suggestion: The corsair book
</message>
```

Proposed framework for commentary in FloWr

```
name:process_node_name
source:parameter_name, type:____
condition:parameter_name,  type:____
output:variable_name, type:____
description:____
```

Example script nodes from FloWr (with their proposed commentary)

```
text.retrievers.SimileRetriever.SimileRetriever_0
fileName:VehicleFacets
descriptionRegex:
aspectRegex:
objectRegex:
lowerEvidencePercent:0
upperEvidencePercent:100
#simileTriples = triples[*].simile
// name:SimileRetriever
// source:fileName, type:file
// condition:lowerEvidencePercent, type:integer
// condition:upperEvidencePercent, type:integer
// output:simileTriples, type:list
// description: search for similes from the file
// VehicleFacets that have an evidence percentage
// from 0 to 100.

text.retrievers.Guardian.Guardian_0
section:all
startDate:2012-01-01
endDate:2012-03-23
numRequired:10
#randomArticle = articles[r1].text
// name:Guardian
// source:section, type:text
// condition:startDate, type:date
// condition:endDate, type:date
// output:randomArticle, type:text
// description: search all sections of
// online guardian articles that date from
// 2012-01-01 to 2012-03-23.
```