Beyond the Creative Species

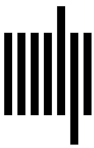
Making Machines That Make Art and Music

By: Oliver Bown

Citation:

Beyond the Creative Species: Making Machines That Make Art and Music
By: Oliver Bown
DOI: 10.7551/mitpress/10913.001.0001
ISBN (electronic): 9780262361750
Publisher: The MIT Press
Published: 2021

The open access edition of this book was made possible by generous funding from University of New South Wales.



The MIT Press

7 Making Creative Systems Effective

We are called to be architects of the future, not its victims.
—Buckminster Fuller

Human nature is not a machine to be built after a model, and set to do exactly the work prescribed for it, but a tree, which requires to grow and develop itself on all sides, according to the tendency of the inward forces which make it a living thing.
—John Stuart Mill[1]

Evaluating the Creativity of Computationally Creative Systems

Empirical Grounding

How do we evaluate whether computational creativity systems are doing creative things, or in the language of distributed creativity, are effective and possibly active contributors in creative production? The radical multidisciplinarity of computational creativity invites a wide range of methods for answering such questions. In engineering and computer science it is often the case that one can specify an objective goal for an AI system. In many creativity-related subtasks this may also be the case. Training a neural network to recognize objects in images requires that we have a dataset of annotated images, meaning that we necessarily have the information required to know exactly how well the system is performing—at least at the given subtask, if not at the wider objective of contributing to a creative task. Meanwhile, human-computer interaction tasks require evaluation of users' responses, and although this is less concrete, data—both qualitative and quantitative—can be gathered about the system's performance. In other

cases, practice-based programmer-artists make work and self-evaluate. In the latter, the evidence becomes vaguer still, but the situation is largely similar to the human-computer interaction context. Systems that generate creative outputs can be investigated using all of these evaluation scenarios and others. A scenario that is somewhat more specific to computational creativity is where people are asked to evaluate the aesthetic quality of the system's output or to otherwise evaluate whether they believe the system to be creative or humanlike.

One theme underlying these discussions is how we establish an *empirical grounding* for various forms of evaluation. Empirical grounding is the establishment of a connection between empirical observations of any form, and the results and conclusions we draw about our systems, traversing a mesh of logic, ontology, epistemology, and language. Statements about systems can take more or less empirically well-grounded forms. Statements of self-evaluation by creators of systems, such as that *the resulting music was interesting*, or that *the system exhibited imagination*, for example, are empirically loosely grounded, whereas to say that *audience members reported the resulting music as being interesting* is empirically well-grounded. Other forms of conclusive remark, such as that *the system achieved human-level performance at creative tasks*, or that *the system was successful because it was the first to be exhibited in a public gallery*, can be equally obscuring of empirical issues, depending on the details of how they are reported and the intention behind their use. They may be weakly empirically grounded, with gaps in the connection between the empirical observation and the point being made.

Such evaluation depends on a commitment first and foremost to understanding, resisting the zealous claims of grant-speak, product marketing, or the pressure to report positive results with every study. There is also a difference between evaluation that results in accurate but ineffective analysis, and evaluation that is fine tuned to contribute to improved designs in an iterative process. Simply knowing whether something was liked or not may not get us very far in terms of knowing what to do to improve it. Evaluation must offer clues as to what can be done better, through thoughtful analysis and good experimental design. This often involves setting up some kind of point of comparison or controllable variable that helps us to break systems down into specific parameters, design features, or properties that can be evaluated independently.

From Turing Tests to Rich Descriptions

Computational creativity evaluation would seem to strike at the essence of artistic subjectivity, and the know-it-when-you-see-it attitude toward creativity espoused by some of the field's founders. Surely we just run these systems, and the ones that produce better outcomes are better? Let the market decide. We can get some way to understanding the achievements of computationally creative systems by subjecting these systems to artistic critique or the popular vote, and some researchers have taken this view. It is reasonable, after all, that a general audience or a panel of experts or possibly the system's collaborative partners, such as fellow musicians in the case of a music performance task, would be able to develop a clear consensus. Recall that in her research Amabile showed that panels of experts did indeed come to a clear consensus on evaluations of creativity and also associated creativity with a clear set of related qualities. Studies of computationally creative systems (such as by Collins and Laney[2] and Lamb, Brown, and Clarke[3]) have also shown significant differences between evaluations by experts and nonexperts. Boden also makes the point that novices may generate the same ratings as experts but may be less adept at explaining these evaluations than experts.[4]

In evaluating creativity it is natural to consider doing studies that take the form of a blind test, akin to the Turing test, where machine outputs are considered in terms of their ability to fool judges into thinking they are actually human outputs. Although appealing, and used effectively in some cases, there are some complications with such an approach. Firstly, in these early days many systems frankly often don't produce great outcomes. Asking people whether they like the output or not might not mean much, and indeed many important advances in computational creativity don't actually come hand in hand with immediate improvements in the generative output of systems. If we are interested in the quality of the systems themselves, then it would be a mistake simply to look at what they produce.

Secondly, as we discussed in chapter 2 with respect to Turing-style tests, it is easy to game this mode of evaluation, or at least, there are many types of distortion or obfuscation that affect how we might understand and approach the evaluation process. The most obvious of these is that we must be able to tell apart the role of the human creator (the programmer, co-creator, system trainer, etc.) in establishing the quality of outputs, leaving the system with a smaller, more manageable task, while giving the impression that the quality

of the output is thanks to the machine intelligence. Consider again the claim, from chapter 1, that a certain system did 95 to 98 percent of the work in composing an album of original music. Such a claim would not only need to be explained but also backed up. Exactly where are human hands used in the process, and how much weight might we give to various production tasks—choosing instrumentation, making minor (or major) arrangement edits, mixing, performance intonation? It would be flippant to take such a distribution of creative attribution at face value. Society devotes much effort to arguing over the minutiae of such issues in human affairs—we may never solve the timeless debate over how much credit the studio engineer, Alan Parsons, deserves for the creative achievement of Pink Floyd's *Dark Side of the Moon*, one of the bestselling albums of all time (in fact, cases such as this are historically responsible for shifting public perceptions about *who* might be considered a creative agent in a particular domain[5]).

In a demonstration of the ease with which *certain* parts of the creative process can be trivialized, James Humberstone, in a TED talk on music, gets the audience to pick notes at random from the chromatic scale and then produces an electronic dance music track from this selection of notes, unadulterated. One point here is that repetition has a massive impact on how you experience a sequence, and an arbitrary bar of music takes on new meaning when repeated several times, setting up reliable expectation and satisfaction. Add some production techniques, with a booming synth sound, and the randomly composed piece begins to sound like a convincing musical work of some genre. There is a conscious sleight of hand here: the assumption that the choice of notes is really the key to the composition diverts us from recognizing that an arbitrary choice of notes can be a fine starting point for a creative process.

This captures part of the problem with what would generally be called an artistic Turing test approach to computational creativity evaluation. But before we slip into this terminology, is it even fair to call the evaluation of a system's output a Turing test, given that it is certainly not a literal application of the text-based imitation test described in Turing's paper?

Writing about generative music systems, Chris Ariza[6] makes the case that the Turing test concept is misused in creative arts contexts frequently. He distinguishes between several different sorts of Turing-related tests. The John Henry Test (JHT), first of all, is simply a head-to-head human versus computer performance test. In general, the JHT does not require blind judges.

Games like chess or Go are domains in which computers have beaten the best human performers, and these have directly measurable and indisputable outcomes. There could be such tests in creative practice, but they need to have an indisputable objective target. For a banal example, we certainly could compare the *speed* at which a computer and human can compose original musical melodies, and indeed this may be of importance in some use-cases. Ariza then classifies the bulk of existing examples of Turing tests in computer music as being "toy" tests, referring to the fact that they do not fully satisfy the requirements of the original Turing test. This builds on Harnad's hierarchy of Turing tests, which views the original Turing test (TT, or T2) sitting in between a total Turing test (TTT, or T3)—the *Blade Runner* scenario, where physiological features are also part of the imitation, culminating in fully humanlike robots—and lesser Turing tests that fall short of the language-based, interactive, open-ended nature of the TT. Those lesser tests, which do not involve free linguistic conversation, are the "toy" tests. Ariza then distinguishes the Musical Directive Toy Test (MDtT) from the Musical Output Toy Test (MOtT). The latter represents the majority of tests that have been used in musical contexts, where people are asked to look only at a system's or human's creative outputs in a blind discrimination task and determine which are made by machines. The MDtT is somewhat more interesting and closer to the original Turing test, in that there is a stage of interaction:

> The interrogator, using a computer interface, sends a musical directive to two composer-agents. One of the composers is a machine, the other, a human. The musical directive could be style- or genre-dependent, or it could be abstract—something like "write sad music," "write a march," or "compose a musical game." … An MDtT could take the form of a real-time musical call and response or improvisation between interrogator and composer agents.[7]

Both, Ariza argues, are not *proper* Turing tests, but at least in the MDtT case there is a level of interaction enabling the interrogator to construct a model of what the system is doing by probing it. Related to this, we can also consider replacing the verbal directives with musical call-and-response and consider how the Turing test might apply to musical interaction in an improvisation scenario. While such call-and-response scenarios are relatively common, it is still more difficult to set up blind tests between people and systems than it is with typed text, and blind tests have been uncommon. The scenario Ariza describes in which verbal instructions are issued is also, until recently, uncommon; given the examples he suggests above,

such systems would require natural language processing abilities and some level of sophisticated open-endedness. Even if a system was very good at writing sad music or writing a march, it would presumably be easy to propose something the system couldn't do that a human could. The systems we have now work over a range of parameters that are usually heavily constrained to a specific domain. But we can imagine more feasible examples from current research, like a parametric creative system that uses sentiment analysis to create outputs that are happy, energetic, or lighthearted. Such a system might have critics agree that it was very good at modeling these qualities but would still be a nonstarter at the *real* Turing test task of understanding advanced musical concepts.

Most tests we see today are of the MOtT form. They are easy to set up and run, and can be designed to result in hard numerical data, such as statistics indicating how many people classed each output in which way. "You be the judge!" compels a feature on the Red Bull website about artificially generated pop songs.[8]

While the MOtT has its place, it is important to look at the nature of its limitations. The outputs alone hide important information about what the system itself is doing. MOtTs have a tendency to invite seduction, to anthropomorphize and exaggerate the capability of the given algorithms, to frame them as humanlike from the outset and then see how far the likeness can be pushed. There is nothing pure about these systems; there are countless hacks and trickery and bias that influence how they produce output. When a GAN-generated artwork sold for over half a million dollars in 2018,[9] surprising the entire art world and the computational creativity community, the work certainly looked like a recognizable painterly style, with original content, but it also seems odd to value a one-off from a system that should arguably be best judged by its mass output. Artworks reach this kind of value because their makers have amassed reputation over a long period of work. This out-of-the-blue exaltation of a particular work of a particular system is somewhat at odds with this.

The system's output can be a poor indicator of the quality of the system because it conceals much relevant information about what type of creative work the system is doing. It is also, arguably, not a particularly relevant way to think about how we judge creativity in humans. We pay great heed to the context of creation. As humans we have a good idea of the cognitive machinery that other humans have at their disposal—it is not something

we have to explicitly question. With humans, also, we can be mistaken in our evaluation; if you discover a new genre of music, then you might consider the first artist you encounter in that genre to be radically original, but you would be acting with limited historical knowledge, not fully aware of the context of creation. Learning about prior influences, for example, might radically alter your view of the originality of a certain artist. Importantly, this is not about whether we like a piece of work, but about how we understand its genesis.

Given the danger of mistaking the generative capability of the system, the situation worsens the more opacity surrounds the information. Thus in the presentation of your computationally creative system you risk entering a *zone of incredulity* when you show only a small number of handpicked outcomes, worse still if they have been tampered with or edited into some more developed work. That's not to say that the outputs of computationally creative systems shouldn't be curated or improved—it depends on the application context. The creators of *Beyond the Fence*, the "world's first computer-generated musical," were well aware that when the first-night curtain opened to a West End crowd they couldn't just serve up the experimental musings of a team of academics and their algorithms; professionals were involved to ensure quality. There is artistic license there, as long as when the behind-the-scenes work is presented it is done honestly.[10] Likewise, I have video examples online of my improvising music system Zamyatin playing live. For the purpose of documenting this work, I choose the best takes, naturally, as I would with any live recorded work. My commitment to academic honesty does not extend to putting the bad takes of my system in my creative show reel (though it does extend to me offering uncurated versions and live demos to fellow researchers, if they are keen to hear them). When computer-generated work is presented to the public it is both a demonstration of a technology and an artistic output in its own right. Thus creators must strike a balance between intellectual honesty and alright-on-the-night production standards. The danger lies in hailing such limited and controlled outputs as a form of evidence, saying, "See for yourself how sophisticated this system is" without being able to back that up with clear information about the role of the system.

That said, MOtTs or their equivalents in other domains can be useful to demonstrate the shortcomings of existing systems. In a National Public Radio listener survey comparing human and machine sonnets, run by Dartmouth's Neukom Institute (discussed below), the listeners' scores showed

that machines were completely incapable of fooling them. The more borderline cases, at 65–35, were those where a human-composed sonnet was mistaken to be computer-composed.

But as Ariza argues, even if the system demonstrates some form of interactivity and open-ended capacity that allows one to probe its behavior, this does not necessarily make it a Turing test, hence the descriptor "toy." In the original test, the question is whether a machine possesses intelligence. The method is a blind game of imitation; it is interactive, linguistic, and open-ended. Marvin Minsky asks what it means for a human or computational system to comprehend that "you can't push a car with a piece of string." What intelligence is involved? Language gives us quite open access to the capacity of another mind, but art, although affectionately seen as a window onto the soul, has a transparency issue. In practice, what has tended to happen is that artistic Turing test results have come back affirmative—yes, this does look like human art/poetry/music—yet leaving us none the wiser as to what is going on. The bar is too low, at least in some areas. Once again, any example from the world of found art, field recordings, or some flavors of mash-up, drone, or noise music might serve this point well; these fields make quite a mess of an output-based Turing test concept. If a system was designed to grab found images from webcams on the Internet and frame them as artworks, I do not think there would be any grounds to reject such works. They must pass.

An artistic variant of the Turing test begs the more basic question: can the Turing test even have variants? The answer, I think, should be no. If the goal of the test becomes split into two distinct goals—(a) be convincingly human-like, and (b) produce good art—then we encounter some conflict between these goals, for which quite different methods come into play. The Turing test was designed to address item (a). For item (b) we should take a more invasive and open approach, and put aside questions of humanlikeness. Ariza summarizes that "the TT is an attractive concept: it is not surprising that it has found its way into discourse on generative music systems. More practical methods of system evaluation, however, may do more to promote innovation."[11]

A more expansive approach would not necessarily reject outright the idea of a blind test. But with the various issues surrounding Turing tests, not least the view that Turing's original use of the tests was as a conceptual thought experiment, not a serious proposal for testing something, the case for blind tests is diminished.

One reason we would perhaps see the need to conduct blind tests, however, is in order to ensure that no bias for or against machine creativity was influencing judgments. Cope was convinced of bias against his machine compositions, and the issue has been raised on other occasions that people judging the work produced by a computer might believe it to be inferior. Taking essentialist views such as Bloom's into account, this is quite plausible. If we care about who made the work, and allow that to influence our view of the work, then of course we may see computer-generated work differently from human creations. This hypothesized bias has been investigated, and the evidence has fallen both for and against its existence in different studies.[12] It remains to be seen if this bias can be definitively shown, but it should be borne in mind that what might seem to be a bias is actually an important part of the puzzle for making good computational creativity systems. Creators do have life stories, personal struggles, relations, cultural and political affiliations that are considered important in the judgment of creative work. It would be wrong to cast off these factors used in judging creative work as mere bias. But as noted by various commentators (such as Colton and Banerji), equally we may be inclined to project these human social properties onto systems even where they don't exist.

A simple solution to the various issues with artistic Turing tests is to expand the approach to data collection and communication. Audience ratings may be important, but they must be augmented with as much relevant information as possible. It may be useful for tests to be blind, but it can also be useful for them not to be. Binary tests (is it or isn't it *X*?) may be less useful than rich descriptions of both the system itself and people's responses to it. Thus we can think of a rich description approach, which draws on interaction design research methods, may use blind or nonblind tests, qualitative or quantitative methods, objective tests and so on. It transcends the binary simplicity of the Turing test. Social scientific methods based on observation and discourse provide qualitative information to support an understanding of how we perceive systems. Dan Stowell and colleagues[13] explore an array of design research methods that they compare to a Turing test approach in a musical context, with a particular focus on discourse analysis and the identification of actors and relationships between them. Information is gathered in the context of more or less structured tasks that reveal interaction affordances and conceptions.

Stowell et al. conclude that Turing-style tests can be useful in certain contexts, but that a number of constraints regulate when this may be the case:

1. *Is the system primarily designed to emulate the interaction provided by a human, or by some other known system?* If so, the musical Turing test method can be recommended.
2. *Is the performer's perspective sufficient for evaluation?* In many cases, the answer to this is yes, although there may be cases in which it is considered important to design an experiment involving audience evaluation. Many of the same methods (interviews, questionnaires, Turing-test choices) are applicable to audience members—and because audiences can often result in large sample sizes compared against performer populations, survey methods such as Likert scales are more likely to be appropriate.
3. *Is the system designed for complex musical interactions, or for simple/separable musical tasks?* If the latter, then simplified tasks may hold some attraction. If the former, then we recommend a more situated evaluation such as our discourse analysis approach, which avoids the need to reduce the musical interaction down to atomic tasks.
4. *Is the system intended for solo interaction, or is a group interaction a better representation of its expected use pattern?* The experimental design should reflect this, using either solo or group sessions.
5. *How large is the population of participants on which we can draw for evaluation?* Often the population will be fairly small, which raises issues for the statistical power of quantitative approaches. Qualitative approaches should then be considered.

Anna Jordanous, whose PhD thesis[14] was the first to focus specifically on evaluation in computational creativity, similarly develops a multifaceted approach that combines qualitative and quantitative methods, including those derived from design and social sciences. She notes that research in computational creativity tends to shy away from evaluation due to conceptual and practical difficulties. A third of the papers she surveyed did not critically discuss creativity, half had no discussion of evaluation, and only a third attempted to evaluate the creativity of the systems they presented. One issue Jordanous identified was the perception that it was difficult to find grounds for comparative studies between different systems, since different systems are geared toward different aims or contexts. "One researcher noted the difficulty of finding 'even one other system that's doing exactly what you're doing," hence causing practical difficulties in comparing systems like for like."[15] By contrast, in some areas such as narrative story generation systems, there were seen to be enough examples of

systems doing similar things, and correspondingly, attempts at comparative research existed in this area.

Jordanous puts forward a flexible approach to evaluation which begins by identifying aspects of creativity that are relevant to the specific domain. From a wide analysis of discourse on creativity, she identifies a global list of fourteen attributes associated with creativity, similar to Amabile's associated criteria (here again, treating the complex concept of creativity from a cluster-concept point of view):

- Social interaction and communication
- Domain competence
- Intention and emotional involvement
- Active involvement and persistence
- Variety, divergence, and experimentation
- Dealing with uncertainty
- Originality
- Spontaneity/subconscious processing
- Independence and freedom
- Progression and development
- Thinking and evaluation
- Value
- Generation of results
- General intellect

Jordanous demonstrates this with a study of four creative music improvisation systems. To do this she recruited participants who had experience of music improvisation as well of computer programming. She asked the participants to discuss the above list of terms associated with creativity and think about the terms in the context of improvisation. From the responses, she ordered the list in terms of those elements most associated with improvisation, giving a domain-specific frame for creativity in that context. The above list is in fact ordered in this way, so the top three elements in the list—social interaction and communication; domain competence; intention and emotional involvement—were identified as being significantly more strongly associated with improvisational creativity than the remainder.

She then asked the participants to rate the four systems on each of the measures, resulting in multidimensional scores, which could also then be weighted according to the respective significance of each of the associated features. Importantly, the participants were not only asked to look at the outputs of the system or interact with the system but also to read about each system and attempt to understand how it worked. The result is a rich engagement with both the system and the nature of the evaluation question being asked. This significantly diminishes the mystique around the evaluation of creativity. Note, for example, that the top three aspects of improvisational creativity identified in the study don't directly engage with the concept of novelty at all.

Looking Under the Hood

The idea of looking under the hood to accurately evaluate computationally creative systems leads to a situation where we may want to more formally break down creative events performed by machines and people into categories. Although creativity theory gives us notions of stages in creative thought such as incubation and illumination, these may be too abstract to associate with specific outputs from a computational process. Forming concepts, developing an idea and implementing that idea are each relevant, semi-independent creative processes, and the evaluation of a system may involve understanding exactly which of these tasks it actually performs. In all creative systems, the system will perform some of these activities independently, but it may also embody its programmers' creative input. It is important to tease apart these factors, but how?

Colton, Pease, and Charnley[16] propose a breakdown into four domains into which creative output might be organized. These are:

- *Framing* Anything that establishes the context for a creative act[17]
- *Aesthetic measure* The measure by which something's aesthetic value is determined
- *Concept* A general schematic for the production of work
- *Example of concept* A specific example of work conforming to a concept

This "FACE" model is then further divided into creative acts that result in actual objects and creative acts that result in processes, which will then be used to create objects. They also propose that along with creative acts, administrative acts need to be considered in the model: those acts that aren't directly creative but that are auxiliary to the creative process.

This breakdown is then applied to the analysis of the creativity of systems, which takes into account both human and nonhuman actors in the attribution of different creative and administrative acts. In conjunction with this, Colton, Pease, and Charnley propose the IDEA model[18] for framing the context of development: "Software is not developed in a vacuum, so we must factor out the input of the programmer/user/audience when we assess the impact of the creative acts performed by software. To do so, we assume an (I)terative (D)evelopment—(E)xecution—(A)ppreciation cycle within which software is engineered and its behavior is exposed to an audience." This allows for situations in which a programmer might change the code of a system (it is therefore properly framed by a distributed creativity point of view). Thus when a programmer writes a generative program, looks at the results, and then modifies their code in order to produce better results—as all practice-based computational creativity artists from Cohen onward have inevitably done—this model has built into it the distinction between what the system does and what the human does, while still seeing the totality as creative.

Instances of creative acts performed by different actors from one of Colton, Pease, and Charnley's analyses include the following:

- The programmer uses their aesthetic to select an image for printing.
- The software generates a new set of functions by crossing over two pairs of functions.
- The programmer takes their aesthetic and turned it into code that can calculate a value for images.
- The software uses machine learning techniques to approximate the programmer's aesthetic.
- The software invents a novel aesthetic function.
- The software applies the new aesthetic to choosing the best image from those produced.
- The software produces a commentary about its process and product.

Although Colton, Pease, and Charnley's FACE and IDEA frameworks offer terminology to aid analysis, it is not universally agreed terminology. The terms *framing*, *aesthetic measure*, *concept*, and *example* risk assuming a certain system architecture or set of relations between individual components. As we have seen, aesthetic appreciation can be multifaceted and endowed with meaning that complicates the notion of it being measurable. But a generalized approach in this spirit, which attempts to identify creative events within

a wider interaction scenario and attribute authorship to those events, can be effective and helps us get away from a limited black-box perspective.

Putting Numbers on Aspects of Creativity

Given a rich descriptive approach to evaluating computationally creative systems, a common scenario that arises is one in which we look not at specific system outputs but at the distribution of outputs that the system produces. Even if this is not done in an interactive way, we can require that a proper system evaluation means seeing a significant quantity of output that is demonstrative of the system's scope, rather than a single output or curated selection of outputs. Such a corpus can be subject to more formal analysis, including the potential that with large enough numbers we begin to build a more statistically grounded view of the system's behavior. This can also be done in the context of another large dataset: the body of existing work in the world that the system has been exposed to. By taking a statistical view over larger numbers of outputs we can begin to characterize and perhaps quantify what type of creative affordances a system might have, building a profile that can be tailored to specific creative goals. This accords with the view that creativity itself has different manifestations and that there may be different ways for us to recognize creativity, as we saw previously with Jordanous's list of creativity-related terms.

Any computational system may be able to produce infinite outputs and might be interactive or data driven, so it may not be practical or possible to consider the entirety of its output, but we can at least aim to go large, or even interactively probe the system, and see what this tells us. If we do so, it may become immediately clear that the system produces work of a high quality but only within a tightly limited range that appears formulaic after sufficient iterations (this is common for generative systems and can often be desirable), or that the system produces mostly poor work, with the occasional, one-in-a-thousand moment of extraordinary happenstance discovery. An important observation to make is that neither is necessarily a failure. Even some highly esteemed artists might be said to exhibit the first trait, quite consciously. And a system of the latter variety might be effective in a domain like still images—it doesn't take that long to flip through 100 thumbnail images in search of certain desirable properties, so this success rate *might* be considered usable in certain co-creative professional contexts. (As Pierre Barreau from Aiva put it, it's not like a self-driving car where a 99 percent success rate is not

good enough. Even a 1 percent success rate can be effective.[19]) It helps to be able to build this kind of profile of our systems, where we attend to different ideas of what creative value a system might have; this is key to understanding what their creative capacity is.

To this end, one of the innovators of the modern field of computational creativity, Graeme Ritchie,[20] set out to formalize the various ways any set of outputs could be described as creative in relation to whatever came before it. In Ritchie's formulation, the stuff that came before is called the "inspiring set." This is stuff the system knows about—in other words, the database of existing artifacts that has been fed as an input to the system.[21] In this formulation it is generally implied, though not essential, that the inspiring set is limited to a single, reasonably well-defined artistic domain, such as jazz music or abstract expressionism. Ritchie then defines intermediary measures for each of the system's outputs. One is *typicality*, which is a measure of how much the output resembles a typical example of the inspiring set. Another is *quality*, a measure of the perceived or otherwise computed quality of the work. The last is *novelty,* which is a measure of how different outputs are from the inspiring set.

It is interesting that Ritchie includes typicality. While on the one hand typicality may be seen as something that is undesirable, being boring or unoriginal, it bears a more complex relationship with value. As we saw in chapter 4, creative works are often created with the aim of having clear membership of an existing domain, thus to be *typical*. This may be as part of an expression of identity and also possibly in order to interact with the perceiver's expectations in managed ways. Something can be typical and novel at the same time. Recall that the shifting nature of conceptual spaces means that any agent within a domain can potentially construct a new set of criteria for understanding the domain, and thus a new dimension along which novelty is introduced, which doesn't upset the parameters of the existing domain. Something that is at once typical and novel clearly displays the traits of a given domain but does something exceptional within those constraints. Typicality is also arguably more easily evaluated than novelty. For example, asking if a piece of music is a typical example of blues can make use of more concrete criteria than asking if it is original, and can also be inferred from indirect data such as annotations.

Having defined these intermediary concepts, Ritchie defines eighteen measurements related to creativity, described as criteria for "attributing

creativity to a computer program."[22] Criterion 1 simply states that the average typicality of artifacts should be above a certain threshold. Thus here typicality is viewed as a basic requirement for the system, to satisfy the expectations of the given creative domain. Criterion 2 is a different slant on the same idea, stating that the proportion of artifacts passing a given threshold of typicality should be greater than a given ratio. Criterion 3 and 4 do the same for quality. Later criteria dig into higher-order notions of creativity, blending these lower-order elements. One of the most common conceptions of creativity is about finding outputs that are atypical and yet still high quality, and in Ritchie's formulations there are several ways we can do this. Criterion 5, for example, states that out of the set of the system's atypical outputs, some given proportion must be high quality.

The full list is as follows:

1. On average, the system should produce suitably typical output.
2. A decent proportion of the output should be suitably typical.
3. On average, the system should produce highly valued output.
4. A decent proportion of the output should be highly valued.
5. A decent proportion of the output should be both suitably typical and highly valued.
6. A decent proportion of the output is suitably atypical and highly valued.
7. A decent proportion of the atypical output is highly valued.
8. A decent proportion of the valuable output is suitably atypical.
9. The system can replicate many of the example artifacts that guided construction of the system (the inspiring set).
10. Much of the output of the system is not in the inspiring set, so is novel to the system.
11. Novel output of the system (i.e., not in the inspiring set) should be suitably typical.
12. Novel output of the system (i.e., not in the inspiring set) should be highly valued.
13. A decent proportion of the output should be suitably typical items that are novel.
14. A decent proportion of the output should be highly valued items that are novel.
15. A decent proportion of the novel output of the system should be suitably typical.
16. A decent proportion of the novel output of the system should be highly valued.
17. A decent proportion of the novel output of the system should be suitably typical and highly valued.
18. A decent proportion of the novel output of the system should be suitably atypical and highly valued.[23]

Ritchie's criteria avoid pinning down a single conception of creativity in a single equation, seeking instead a diversity of mathematical formulations of varying ideas about what creativity might mean. As with Jordanous's list, from a family resemblance point of view we can see how all of Ritchie's criteria resemble some sort of typical comment a person might make about someone else's creativity: *he really understood the style and was able to take it further*; *she didn't do radical work often, but when she did it was brilliant*.

How useful this is in practice depends firstly on how well and how easily we can determine typicality, quality and novelty. The default way is to ask people, perhaps in large numbers via an online survey. In certain circumstances it may also be possible to automate this, although we have seen the problems with the automated measurement of quality. Besides, if the system has access to the same process to evaluate typicality and quality as the automated analysis uses, then the problem can be considered largely circular! It is also challenging to think about applying Ritchie's ideas in a non-domain-specific context, where typicality may be harder to define.

A multiplicity of measures for defining the creativity of systems sits well with the rich description approach to evaluating computational creativity and the family resemblance stance. It also feeds well into applied areas of computational creativity, with the various qualities identified by Ritchie's criteria relating to different use-cases, domains, and stages of the creative processes. Systems made for concept ideation might produce short, rapid-fire, multiplicitous suggestions and might have low typicality requirements, with parameters to control the breadth or level of risk involved in the output generation. Systems made for live musical accompaniment would need to work in real time without feedback or curation and so might have much stricter typicality requirements.

On the typicality front, another important consideration is plagiarism. We want things to be typical of a style or even of a specific artist, but we rarely want them to fall into the plagiarism category. This is a fine legal line. Asked in an interview if Emmy ever plagiarizes, Cope responds: "I'll let the courts decide this. … Note, however, that most composers plagiarize dozens if not hundreds of works in a single piece of music, and most do this subconsciously."[24] Cope is alluding of course to the fact that copying style is a critical and functional part of creating original music. Yet most creative work, derivative though it may be in various respects, manages to clearly avoid accusations of plagiarism, which in specific creative domains have clear and

known parameters. Systems such as Aiva run plagiarism detectors to double check that their output doesn't end up reproducing verbatim sections of the input music, which is perfectly possible in the context of RNNs regurgitating transition expectations. Unlike the measures we've talked about above, plagiarism should be a more hard and fast thing to check for, depending on known and legally argued rules dictating what counts; generative algorithms should be able to easily run their output through plagiarism checkers.

In one final example, Elgammal and Saleh[25] conduct a study to attempt to quantify creativity in human-produced work. They do this using an automated process running over a massive dataset of Western artworks since 1400. What is interesting in this study is that they measure creativity using a standard combination of novelty and value, but they measure value as *influence*, which is measured with hindsight with respect to the works that followed. That is, if one artwork can be shown to have influence on future artworks, its creativity score goes up. This is a fascinating idea, but entirely concordant with an evolutionary, teleofunctional, generative perspective in which the value of works is determined by their ability to stick around and cast influence.

Running Prizes

A number of competitions have been run to try to identify successful systems. I visited the Neukom Institute at Dartmouth College, based in New Hampshire in the US, just as they were judging their Neukom Turing tests for the creative arts in June 2017. The prize was in its second year and had four separate tasks, one in music, one in dance (applied to the animation of 3D figures), and two in literature. The AccompaniX accompaniment task required the system to follow a musical melody, also with a score provided, and come up with a musical accompaniment. The DanceX task required the system to control the limb coordinates of a 3D virtual dancer duetting with the recorded data from a real dancer. The DigiLit task required the system to complete a short story seeded with the first half of the story. And the PoetiX task required the system to generate sonnets based on a single word theme fed to it.

An ambitious thing the Neukom team did was to require that participants submit working programs rather than simply outputs, running those programs using novel inputs to establish a basic degree of interactivity with

the systems. The prize committee organized creating the test inputs, running the programs to generate various outputs, and then having a judging team, including the general public, judge those outputs. In this way, they combined a blind MOtT (or equivalent in other domains) with a degree of interactivity in that the systems have to be able to work with unknown inputs. Attempts to look inside the systems or to generate large batches of outputs were beyond the scope of this team's capability; although this is positively argued for in the above work, it is hardly practical to do so with every test.

Similarly, each task was designed so as to have a reasonable level of common interest, to be carefully constrained to a domain-specific area, and to allow interaction. In other words, the tasks had to be widely appealing, manageable, clearly defined and yet still challenging and capable of demonstrating clear creative capability. In areas such as music, being widely appealing, while at the same time being sufficiently domain-specific as to state a clear goal, is a considerable challenge. The team had to avoid highly genre-specific tasks so as to widen the field but also to avoid something so completely genre-ambiguous so as to have clearly defined objectives.

In each case, human responses to the tasks were also gathered, so that a blind user test could take place. Some interesting discussion arose here around the poetry task. For example, use of punctuation and formatting such as italics had to be removed from the human-composed poems because this was a clear giveaway, with the computer-composed poems generally avoiding such elements for simplicity and greater success. This prompted the question of whether dumbing down the human work by constraining it in this way was an acceptable thing to do in order to create a level playing field.

Related to this was the question of whether the poetry should be constrained to a narrative form. Non-narrative poetry is much easier for computers simply because it potentially removes a level of advanced concept manipulation which is harder for existing algorithms to manage. Poems that riff on a series of associations in an impressionist manner without the burden of constructing a meaningful narrative are much easier to get a program to produce. In the world of human poetry, there is nothing invalid about such works, and yet the fact that this makes it easier for computer programs prompted some people to suggest that perhaps the call *should* include the requirement to work in a narrative form.

Practice-Based Artistic Research

Practice-based research is research in which a creative practitioner engages in their practice in a reflective and systematic manner, with a mind to making contributions to knowledge in an academic forum. Knowledge outcomes may take the form of knowledge about the processes of creating work, such as the methods by which a painter chooses a color palette, or knowledge about the materials and products themselves, such as how color exposure achieves persistence effects and complementary colors in, for example, the light works of James Turrell. Linda Candy[26] distinguishes these with the subtly different terms practice-led (about the practice) and practice-based (about the outcomes) research, although it is common to see practice-based research used as a catch-all term for simplicity. As a celebrated example of the power of practice-oriented contributions to knowledge, consider Barbara Bolt's[27] discussion of the artist David Hockney, who researched the use of visual aids in historical drawing practice "in which Hockney literally draws out his emerging understanding through this practice and his long-standing immersion in it."[28] Hockney was able to identify the use of visual aids by historical artists because of his intimate firsthand experience as a drawing artist and was able to experimentally verify these ideas through his own practice. We can see immediate parallels with the story of Harold Cohen and his expert development of AARON's drawing and coloring skill, and the similar stories of David Cope and George Lewis.

Practice-based research is an important dimension in the evaluation of computational creativity systems simply because much of the work in computational creativity has historically been done by practicing artists. More generally, it is an effective and efficient setup: an individual creative practitioner, working with code, incorporates creative autonomy into the software they are making, deploys it in real creative scenarios, and then has the potential to evaluate the system using a variety of means, adding individual reflection to other formal methods such as audience or performer interviews and surveys. As discussed in chapter 1, this works well because they come with both the expert knowledge of their domain that enables them to develop the system and the social connection to the domain that enables them to create the contexts in which the work is situated.

An artist reflecting on their own work risks a lack of objectivity that those working with managed user studies strive to avoid. But it can easily

be forgotten that if they apply rigor and a commitment to intellectual honesty, they can achieve much of what might be achieved by a much larger study. Even formal studies that have one or two third-party users can be highly effective at identifying core themes and design issues. In user interface design, Nielsen and Landauer showed that as the number of users tested increases, the number of problems discovered plateaus relatively quickly.[29] In a trial of user tests revealing design problems in a mock system, they found that even a single user test could reveal up to a third of the potential problems. Three users revealed well over half the problems, and adding further user tests resulted in diminishing returns. Above ten users, very little was gained by adding extra users. Much like Knuth's maxim that optimization is often not worth doing ("We should forget about small efficiencies, say about 97% of the time: premature optimization is the root of all evil"),[30] an obsession with formal studies can be a poor use of resources. This is not to say that statistically significant results with larger sample sizes can't be valuable in other circumstances. Thus when it comes to working creatively with autonomous systems, taking a practice-based or autoethnographic approach can simply be a quick way to get meaningful results that inform future design decisions.

In practice-led or process-focused research, evaluation takes the form of knowledge about the individual's creative practice and how the system's creative autonomy can best be put to good use. Cohen, Cope, Lewis, Brown, and other early creative practitioners can be seen to be engaging with this form of knowledge gathering, even if they do not explicitly identify it as such. This includes philosophical reflection on their work, which potentially stimulates new concepts. Cohen's discussion of the creative autonomy of AARON in chapter 1 was of this nature.

A generation of electronic media art researchers now combine computing research with practice-based research, including HCI and interaction design methods where necessary. These practitioners develop original creative works involving innovation in algorithms and methods. This leads to the potential for knowledge outcomes across three levels. Formal studies of the algorithms objectively demonstrate algorithmic properties. For example, a generative system might be formally analyzed using any of Ritchie's criteria. Practice-based studies are subjective and address the artist's workflows and methods. Other HCI-oriented studies address third-party users, including co-creators and audiences.

For example, in studies by Jon McCormack and myself[31] on the use of ecosystem computer models as creative tools, we reported on the formal properties of those ecosystem models, the creative potential of using these models, and we outlined candidate design principles that could be used to create creative tools for third-party users.

In my own work on autonomous music improvisation[32] systems and in related work by Yee-King,[33] Eldridge,[34] Vear,[35] Banerji,[36] Ackermann,[37] Carey,[38] and others, studies have been conducted with very small numbers of musicians (between one and ten, usually around two or three), performing with the systems and engaging in detailed discursive analysis. This has included focus groups, discourse analysis, tests based on dedicated tasks, and so on. In my work with Zamyatin, for example, I found that performers had a greater sense of system autonomy when playing with the system than when watching someone else play with the system.[39] This reinforces the idea that looking at outputs is less informative than an interactive paradigm. I also found that performers were able to use sophisticated language that did not fall into anthropomorphization of the system—they definitely did not view the system as a virtual musician—while being able to discuss and describe the system character. Performers intuitively wanted to prod the system through performance with it, to gain an understanding of the system, whether a formal descriptive understanding or a more intuitive sense of what it would and wouldn't do. Through this process they habituated to the system and quickly became comfortable playing with it. In this sense one could see the system as falling somewhere between performer (albeit minimal in its humanlikeness), instrument, and composition.

Interaction Design for Computational Creativity

I began this chapter discussing empirical grounding. While the empirical grounding of systems producing cultural artifacts in opaque ways is relatively challenging, it is easier to empirically ground the evaluation of systems in operation-based and co-creation contexts where there is a clear human-computer interaction scenario established, and the system can be seen to perform certain functions. Thus we can look at evaluating the success of computational creativity systems from the point of view of an interaction designer, picking up themes from chapter 6 and the various application scenarios considered there.

Supporting Artistic Creativity and Open-Ended Exploration

Creative tasks have formed a distinct strand of focus for interaction designers, creative technologists, and educators for some time.[40] As well as simply getting known tasks done efficiently and accurately, creative spheres must support the conditions for creativity, which may include phenomena such as managing broad searches, a sense of flow, the ability to handle complexity, and the ability to precisely manipulate artifacts in multiple ways. The design of creative software tools has responded to this specific set of needs. Resnick and colleagues[41] describe the most general requirements for creative software systems as having *low floors, wide walls, and high ceilings*. Low floors refers to the barrier for entry. Software should aim to be as learnable as possible, a benefit to its adopters and also clearly quite valuable to the software's creators. Wide walls refers to the breadth of the software's capabilities. It needs to be versatile and allow for a wide diversity of outputs. Individuals should be able to carve out their own unique forms of expression that are facilitated rather than dictated by the software. This requirement is interesting because creatively empowering *constraints*, factors that narrow the space of creative possibilities and intensify focus, can also be something that creative practitioners desire.[42] High ceilings refers to the potential richness or complexity of creative reach made possible by the system. This is particularly relevant in computational media where creators can develop outputs of great cumulative complexity, through programming software behaviors, for example. More generally, this requirement is about being as enabling for professional users as for beginners.

Different design factors can support each of these requirements. Software that instructs you as you work, provides built-in help, enables you to get started by hacking existing examples, or that is designed so that the initial interface is stripped down and focused on more elementary tasks can support the more rapid adoption of the software and the achievement of basic goals (low floors). This supports the constructivist learning principle that you learn best through doing, particularly through exploring and discovering things from your own mistakes. It is also encouraging and thus enjoyable for users, and of course means that the user is being rewarded with creative outputs from the outset, and achieving their objectives in the shortest possible time with the least effort involved in learning. Csikszentmihalyi's principle of *flow* informs how we might think about scaffolding further levels of complexity and learning experience.

Designing for wide walls and high ceilings can be implemented in many ways: providing different tools and perspectives on the same artifacts; enabling a plug-in or library-based community approach that enables the software to be expanded endlessly; providing generalized rather than specific solutions to user needs; enabling different input and output formats and interoperability with other software; and providing professional functions such as unlimited undo and batch processing.

Effectively Communicating with Computers: Language, Code, and Graphical User Interfaces

In imagining how we might support these objectives of creativity support tools, we can take stock of two extremes that stand as alternatives to traditional graphical user interface (GUI) based control of digital systems, one where we talk to machines in much the same way that we communicate with other people, and the other where we use code as the core medium for computational expression. The former is, on the surface, more inclusive and requires little training or expertise, at least in the operation of machines, but remains an emerging technology, and it is not clear what kinds of intermediate forms we will have to work through as we get there. The latter suggests a high barrier to entry, and yet is widely recognized as the best way we currently have to interact with the full scope of computational intelligence available to us. As coding becomes easier and coding education becomes more focused in schools, the maxim that "coding is the new literacy" may play out. We already inhabit a rich ecosystem of programmers, from those making new chip designs and working in assembly languages to creative professionals who occasionally script something or write a macro in Excel.

Although it is not the only way to create complex computational outputs such as generative and interactive works, artists who work with code are more readily able to access the tools needed to achieve these aims; they can either create them themselves or incorporate existing programming libraries into their work in ways that cannot be achieved by nonprogramming end-users. Code is an eminently shareable medium and with the strong culture of open-source software one can access large repositories of existing code that has been made to perform specific functions, including supporting interoperability with external hardware. Thus in terms of wide walls and high ceilings, code is extraordinarily powerful. The problem for creative coding has tended to be the low floors dimension. Code is tricky

to learn and can be incredibly taxing and frustrating to use. While many efforts are focused on making code easier, the vast infrastructure of the digital world also involves a certain amount of lock-in, where complexity and opacity protect expert knowledge.

Frameworks exist for helping us formalize these usability issues. Blackwell and Green's[43] cognitive dimensions of notations framework, for example, applies to digital manipulation tasks and is generally useful in identifying bottlenecks in usability "to analyze the interactions between: a) the structure of the information, b) the environment that allowed that structure to be manipulated, and c) the type of activity the user wanted to perform."

They identify the following set of cognitive dimensions affecting how one can go about performing a complex task with a piece of software:

Viscosity Resistance to change.

Visibility Ability to view components easily.

Premature commitment Constraints on the order of doing things.

Hidden dependencies Important links between entities are not visible.

Role-expressiveness The purpose of an entity is readily inferred.

Error-proneness The notation invites mistakes and the system gives little protection.

Abstraction Types and availability of abstraction mechanisms. Systems that allow many abstractions are potentially difficult to learn.

Secondary notation Extra information in means other than formal syntax.

Closeness of mapping Closeness of representation to domain.

Consistency Similar semantics are expressed in similar syntactic forms.

Diffuseness Verbosity of language.

Hard mental operations High demand on cognitive resources.

Provisionality Degree of commitment to actions or marks.

Progressive evaluation Work-to-date can be checked at any time.

With viscosity, for example, the question is how easily one can make changes, including going back to change something they've done. A typewriter is particularly viscous compared to a modern word processor; it is difficult to change text that has been written. With provisionality, the question is how easily someone can play with a set of possibilities without making commitment to those possibilities. Wherever computationally creative systems need interfaces, these kinds of straightforward usability questions come into play.

Natural language-style programming languages have been elusive. Languages like Apple's AppleScript aspired toward natural language-style

syntax, but although this might have improved readability it did not make the programming experience any less brittle, requiring a precise and limited use of words and syntax ordering as usual. But the intense demand for natural language processing systems now means that we have far more fluid interfaces that appear poised to enable programming-like actions by non-programmers, as well as programmers operating with objects, behaviors, and data they are not familiar with. This book is written at a possible cusp in this field where submitting requests to a computer via natural language is now relatively normal but has not yet opened up the capacity to operate a machine with the open-endedness of programming via natural language. It may be a short while before it is possible to issue open-ended instructions in the form of requests for content generation to a system that would in turn summon generative functions to satisfy that request. With the convergence of a range of AI-based technologies such on-demand generation could arrive very quickly, transforming the basic nature of the interfaces with which we perform computational creativity, perhaps enabling a far more dynamic and less brittle interaction with computers in general. But the technical challenges of building and integrating systems broad enough to satisfy open-ended requests should not be underestimated, and as ever the question of whether such means of interaction with machines will be embraced by a population of creative users remains dependent not only on the capability of the technology but on the cultures of practice surrounding it.

The open access edition of this book was made possible by generous funding from University of New South Wales.

This book was set in Stone Serif and Stone Sans by Westchester Publishing Services.

Library of Congress Cataloging-in-Publication Data

Names: Bown, Ollie, author.
Title: Beyond the creative species : making machines that make art and music / Oliver Bown.
Description: Cambridge, Massachusetts : The MIT Press, [2021] | Includes bibliographical references and index.
Identifiers: LCCN 2020017931 | ISBN 9780262045018 (hardcover)
Subjects: LCSH: Technology and the arts. | Arts—Technological innovations.
Classification: LCC NX180.T4 B68 2021 | DDC 700.1/05—dc23
LC record available at https://lccn.loc.gov/2020017931

10 9 8 7 6 5 4 3 2 1