Assessing Creativity

Graeme Ritchie
Division of Informatics
University of Edinburgh
80 South Bridge
Edinburgh EH1 1HN
g.d.ritchie@ed.ac.uk

Abstract

In exploring the question of whether a computer program is behaving creatively, it is important to be explicit, and if possible formal, about the criteria that are being applied in making judgements of creativity. We propose a formal (and rather simplified) outline of the relevant attributes of a potentially creative program. Based on this, we posit a number of formal criteria that could be applied to rate the extent to which the program has behaved creatively. A guiding principle is that the question of what computational mechanisms might lead to creative behaviour is open and empirical, and hence we should clearly distinguish between judgements about creative achievements and theoretical proposals about potentially creative mechanisms. The intention is to focus, clarify and make more concrete the debate about creative programs.

1 Introduction

The goal of this paper is to set out some of the issues relevant to assessing whether a particular computer program has been, or is being, "creative". There is no empirical work presented here: the aim is solely methodological.

The question of how a computer program might function creatively is of interest within artificial intelligence. It would be possible to hypothesise that various types of processing (e.g. meta-level reasoning, evolutionary algorithms) are well suited to creative computation. However, any such mechanism has to be judged on how successful it is, as there is no direct way to assess the inherent creativity of a computational mechanism. To judge the (creative) success of a program, we have to have a clear notion of what we mean by a program "being creative". Without that, discussion of the pros and cons of different internal mechanisms are just speculation. This paper tackles that logically prior question of judging the creativity of a program.

We will suggest some formal characteristics of a program's output, and of its construction, which are pertinent to the assessment of its creativity, thus offering a step towards a descriptive model of such assessments. Our framework allows for judgements to be relative to different points of view (either about the merit of various kinds of output from the program, or about the extent to which different factors contribute to creativity), and emphasises how the program was constructed, and the properties of its output, rather than how the program operates internally.

Attention is limited here to programs which produce some set of artefacts (e.g. pictures, stories, conjectures) which are capable of being assessed by human judges in isolation from the process which produced them. Systems which compute abstract entities such as analogies are covered only if the entity (e.g. the analogy) either is used to generate artefacts, or can be stated in some way that is open to evaluation by humans.

2 Creativity judgements

Judgements of creativity originated as assessments of human behaviour, and our whole notion of "creative" is derived from ideas about human creativity. Unfortunately, these ideas are not only imprecise and unformalised, they tend to manifest various prejudices which can impede their transfer to the evaluation of computer behaviour. In our culture at least, certain activities are assumed to be more creative than others. Painting a picture, writing a poem, or creating a sculpture are often deemed creative, even when performed in an ordinary or mediocre manner. Mathematics, science, or engineering are rarely classed as creative, unless they are done exceptionally well. This bias does not seem helpful in a rigorous attempt to pin down the notion of creativity, particularly when applied to machines. Although there is still a tendency within AI to tacitly accept this intellectual apartheid of creative vs. non-creative activities, it would be better if we could be more neutral in our formal characterisation of creative actions. In the rest of this paper, the illustrative domains mentioned will typically be areas such as poetry-writing or story-telling, but this does not mean that the formal definitions or substantive proposals relate only to those activities. We will abstract away from the particular genre of activity, and discuss only the formal properties of the process.

Boden (1992) makes the important distinction between *H-creativity* (producing an idea/artefact which is wholly novel within the culture, not just new to its creator) and *P-creativity* (producing an idea/artefact which is original as far as the creator is concerned, even though it might have been proposed or generated elsewhere in the culture, perhaps much earlier in history). Boden points out that when studying the process of being creative (within a single human or in a computer program) it is *P-creativity* that is at issue, since we are interested in how a single agent can come up with something that is novel *relative to its initial state of knowledge*. A P-creative discovery may prove to be of little use to society because it repeats something that was already known, but that does not render the intellectual or artistic feat of producing the idea/artefact uncreative, viewed in isolation. The mechanisms of creation are what we are interested in here.

When people do assess human creativity, they take into account various factors, and sometimes confuse different notions of creativity (H-creativity and P-creativity, for example). Leaving aside the prejudice in favour of "art" against "science" alluded to above, perhaps the main criterion which is deployed is the following:

Novelty: *To what extent is the produced item dissimilar to existing examples of that genre?*

Closely related to this is a test of whether the item has "artistic" value:

Quality: *To what extent is the produced item a high-quality example of that genre?*

There may be subsidiary tests of creativity, such as whether the person worked unaided, but they are relatively minor compared to the two tests listed above. If a person produces a painting which is radically different from previous work (their own work and work they have seen, for the purposes of considering P-creativity), and which is definitely a good painting, then that will usually be deemed creative. What is rarely brought into the assessment is *how* the person came up with the idea/artefact, that is, the thought processes by which the result came into being.

In the main section of this paper, we shall suggest that when it comes to assessing the creativity of a computer program, the situation is slightly more complex, but also (in principle at least) more amenable to formalisation. Nevertheless, judgements about human creativity are the starting point, as they delineate the intuitive concept. Our proposals will attempt to reflect as faithfully as possible that pre-theoretic notion.

3 The role of computation

Boden (1992) presents a wide-ranging, subtle and thought-provoking discussion of creativity. It is outwith the scope of this paper to summarise and debate her ideas in detail, but it is important to deal at least briefly with one or two of her central points.

Boden argues that true creativity results from *the transformation of conceptual space*. The evidence for this is that a creative achievement produces an artefact (abstract or concrete) which not only is significantly different from previous exemplars, but also establishes new norms by which further exemplars may be classified and judged; for example, early Cubist paintings were not only radically different from previous representational art, they set up a new range of artistic possibilities. She goes on to use this analysis in terms of conceptual space as a criterion for assessing the creativity of programs, by applying it not merely to the final artefact (which could be assessed in terms of similarity measures or the presence/absence of particular properties), but to the manner in which the result is produced. That is, she regards an *explicit transforming of the space* as a *necessary* condition for a program to be creative. This constitutes an interesting step in the argument, and one which it is easy to overlook. It defines the threshold for creativity in machines as being higher than that for humans. When a human produces some novel artefact, the extent to which it establishes a new or transformed conceptual space has to be judged largely from the inherent properties of the artefact. For computers, Boden narrows the goalposts by demanding not only that the artefact exhibit suitable properties, but that the process that gave rise to it (which is much more accessible in the computer case than in the human situation) must meet the test of space-transformation. This also means that a particular theoretical analysis of what might produce creativity becomes part of the empirical judgement of whether a program has been creative.

Boden is not alone in arguing that the manner of computation is relevant to the judgement of creativity, as it is not unknown for critics of AI to refuse to accept programs as creative (or intelligent) once the mundane mechanistic nature of the inner workings are revealed. (See also Section 9.3 below.) In the framework presented below, we will adopt a slightly different position. We take it that the question of which computational mechanisms result in creativity is open and empirical. What we need to do is set down criteria for a program "being creative" which are, as far as possible, based on the ordinary intuitive notion of creativity and which assume as little as possible about theories of how to compute creatively. Then any proposed creative mechanism (e.g. analogical reasoning) can be assessed as to whether it does indeed result in programs which fulfil the independent criteria of creativity.

4 Overall framework

Our aim here is to define as precisely and as formally as possible the attributes one might look for in order to decide whether or not a particular computer program had behaved creatively. As we are attempting to generalise across many possible avenues of creativity (ranging from music to poetry to mathematical conjectures), we need to establish what sort of system we are considering. In sense, what we need is a *reference architecture* (Cahill et al., 1999) which presents a consensus or stereotype of a creating program, and relative to which we can frame our definitions and discussion.

4.1 Basic data

A creating program operates to produce artefacts in some medium. The "medium" is essentially the output data type of the program. At the level of abstraction adopted here, we can ignore (at least provisionally – see Section 9.4 below) the internal structure of the entities that the program produces, and simply postulate a set, possibly infinite, of *basic items*. This is *not* a definition of what would count as a "successful" or "valid" output for the program, merely a statement of the of data type which it produces (e.g. strings of words, arrays of pixels). For example, the basic item set for a program intended to produce simple puns might be the set of a finite sequences of words.

4.2 Rating the output

We want any assessment of items produced by a program to be as faithful as possible to the two notions of **Novelty** and **Quality** stated in Section 2 above. However, those two informal statements tended to assume that the item was indeed a valid example of the chosen genre, which could then be rated for similarity or quality. In the realm of computer generation, life is more complicated, and we must lower our sights accordingly. An implemented system is sure to generate basic items, but there is no guarantee that these will be to any extent exemplars of the intended (and perhaps fuzzily defined) target class. A would-be poetry generator will normally generate strings of words and punctuation (i.e. basic items), but it may be that not all of them are poems, let alone good poems. Colton (personal communication) has pointed out that the HR program (Steel et al., 2000; Colton et al., 2000) produces only well-formed items of the target class – mathematical conjectures – but such tidiness is not universal for generating programs. In computational generation of artistic artefacts, simply managing to produce something which is a valid story/poem/joke etc. is a non-trivial achievement. Hence our formalism assumes slightly less than the two yardsticks from Section 2, and starts at a more fundamental level in its ratings.

We shall take basic items as being *possible* instances of the intended class of artefacts. More subtly, they may be instances to some degree. We will therefore represent a class of artefacts (the target of the creative exercise) as a mapping from the basic items to the interval $[0, 1]$. (This is equivalent to treating the class as a fuzzy set, but that perspective will not be developed here.)

This takes one step towards allowing us to capture the **Novelty** criterion. We will decompose the intuitive idea of novelty (from the viewpoint of P-creativity, which is what we are concerned with here) into two separate factors. Firstly, items which gain low scores on the mapping which characterises the target class of artefacts will be deemed to be dissimilar to the norm for that class. That is, we assume that this mapping encodes the notion of established norms for the artefact class, so that high-scoring items are very much part of the class, and low-scoring ones are implicitly dissimilar from the past practice (in society or culture) which has established the class. Secondly, in Section 5 below, we shall try to formalise the notion of a program producing items which are different from those which guided its original construction. (See also Section 9.6 below).

A useful distinction can be made between properties which measure to what extent an item meets the criteria for membership in the intended artefact class (is it a poem/joke/conjecture/etc.?) and further properties whose presence indicate that the artefact is a *good* instance of this type of artefact (a good poem, a funny joke, an elegant or profound conjecture, etc.). This latter evaluation will also be formalised as a mapping from basic items to $[0, 1]$.

This attempts to capture the second informal property, **Quality**, in Section 2 above.

These two mappings – for class membership and quality – may themselves be based on further definitions (e.g. a checklist of properties, perhaps with weightings attached). At present, we have no firm proposals on what this information should be, but we shall call it a *rating scheme*, and list it separately so that the distinction can be made in our later definitions, abstracting away from its internal details. We shall also assume an operation *APPLY* which, given a rating scheme, creates a mapping to $[0, 1]$. Notationally, we shall usually make the abbreviation of using the name of a rating scheme as if it were the function which APPLY would create; that is, writing `rat(X)` as short for `APPLY(rat)(X)`.

The set of possible rating schemes for a set A will be written as '$\mathcal{RAT}(A)$'.

4.3 Ratings, weightings and spaces

Certain forms of rating schemes would allocate the basic items to points in a multi-dimensional space, which in turn would lead naturally to a measure of distance between items. This might have further advantages in discussing the properties of program output (cf. Section 3 above).

Given the set of basic items, there will be some *properties* that these items may or may not have; or to be more subtle, that these items may have to varying degrees. For example, a poem is a sequence of words that has properties such as – depending on your view of poetry – rhyme, rhythm, imagery, etc.

Definition 1 *Given some set A a **property rating scheme** for A is a tuple $\langle f_1, \ldots, f_n \rangle$ of functions from A to the closed interval [0,1].*

Intuitively, in this definition each f_i represents a property that an element of A may have to some degree. The ordering on the mappings $f_1, \ldots, f_n$ is arbitrary, and it would have been possible to formalise this as a set, but using a vector has the possibly useful side-effect that a property rating scheme (or a weighted property rating scheme, below) induces a measure of similarity on the set A, since the scheme will assign to each item a vector of values, for which a formal distance measure can be defined.

Definition 2 *Given some set A a **weighted property rating scheme** for A is a tuple $\langle (f_1, w_1) \ldots (f_n, w_n) \rangle$ where each f_i is a function from A to the closed interval [0,1], and each w_i is a numerical **weight**, with $\sum_{i=1}^{n} w_i$ = 1.*

This enhanced definition would allow for properties to reflect relative values placed on them by who or whatever is judging them.

In what follows, nothing depends on the rating schemes being property-based, although that would be a plausible form to use.

4.4 The objects generated

We can now use a rating scheme as representing a class of basic items.

Definition 3 *An **artefact class** consists of a set $\mathcal{B}$ of basic items and a rating scheme for $\mathcal{B}$.*

Here we are using a single rating scheme to capture both inherent, measurable properties of a basic item, such as syllable counts, and more subjective aspects. In particular, discussions of creativity sometimes argue that the *expectations* of the audience are relevant – an artefact which exceeds or violates the expectations of the perceiver may be rated more highly. Here, those aspects are packed into the notion of an artefact class, on the grounds that expectations are in a sense a subjective notion of what typifies a particular genre. This should suffice at least as a first approximation.

We shall return to the issue of subjectivity later.

As noted above, we also need a rating scheme to represent the quality of the generated artefact.

Definition 4 *A **value-based artefact class** consists of a triple $(\mathcal{B}, typ, val)$, where $\mathcal{B}$ is a set (the basic items) and typ,val $\in$ $\mathcal{RAT}(\mathcal{B})$ (the **typical ratings** and the **value ratings** respectively).*

We will postpone showing how this formal apparatus can be used to describe creative effects until we have defined in more detail what constitutes a generating program.

4.5 The program

The origins of a generating program are pertinent to assessing its creativity, as is often acknowledged by worries about "results being pre-programmed in". We propose an abstract stereotype of what a creative program is, and how it comes into being, as follows.

From the set of basic items, and taking into account the typical ratings and value ratings relevant to the artefacts of interest, the designer first carries out *selection*, which sets aside some subset, usually finite, of basic items which are to guide the construction of the creative program. We will call this set of basic items the *inspiring* set. There is then a process of *program construction*, which can be viewed as a mapping from the inspiring set (and the relevant ratings schemes) to a program. This construction will often be done informally by the human designer/programmer, but could in principle be automated in some way. We idealise the program as consisting of a *generating procedure* (the main algorithm) and a definition of the ranges of *initial data values* (the possible parameters for the algorithm). The latter definition will be a tuple of sets, each set being the range for one parameter to the generating procedure. (It would be possible to decompose the generating procedure into a set of "rules" and a "generating algorithm", but that level of detail is not needed here, though it might become necessary in some more refined analysis). The generating procedure is then a mapping which, given a tuple of initial data values, produces a set of basic items. To simplify matters, we will assume that the generating procedure is *total* across all the tuples of data which lie within the defined ranges, so there is no question of invalid input values. We also assume that running the program produces a *set* of basic items, rather than a single item. (See Figure 1 for a sketch of the overall scheme.)

Notation: We will use '$\mathcal{P}(X)$' to denote the powerset of the set X.

Definition 5 *Given a set of basic items $\mathcal{B}$, a **generating program** for $\mathcal{B}$ consists of a pair $(\langle D_1, \ldots, D_k \rangle, G)$ where $\langle D_1, \ldots, D_k \rangle$ is a k-tuple of sets, and G is a mapping from $D_1 \times \ldots \times D_k$ to $\mathcal{P}(\mathcal{B})$*

In the above definition, each D_i is the domain for some parameter for the procedure G.

Definition 6
*Given a value-based artefact class $(\mathcal{B}, typ, val)$, a **program construction scheme** consists of a pair $(S_{\mathcal{B}}, C_{\mathcal{B}})$ where $S_{\mathcal{B}}$, the **selection** process, is a mapping from $\mathcal{RAT}(B) \times \mathcal{RAT}(B)$ to $\mathcal{P}(\mathcal{B})$, and $C_{\mathcal{B}}$, the **construction** process, is a mapping from $\mathcal{P}(B) \times \mathcal{RAT}(B) \times \mathcal{RAT}(B)$ to the set of generating programs for $\mathcal{B}$.*

Hence, with the notation used in these definitions, $C_{\mathcal{B}}(S_{\mathcal{B}}(typ, val), typ, val)$ is a pair $(\langle D_1, \ldots, D_k \rangle, G)$

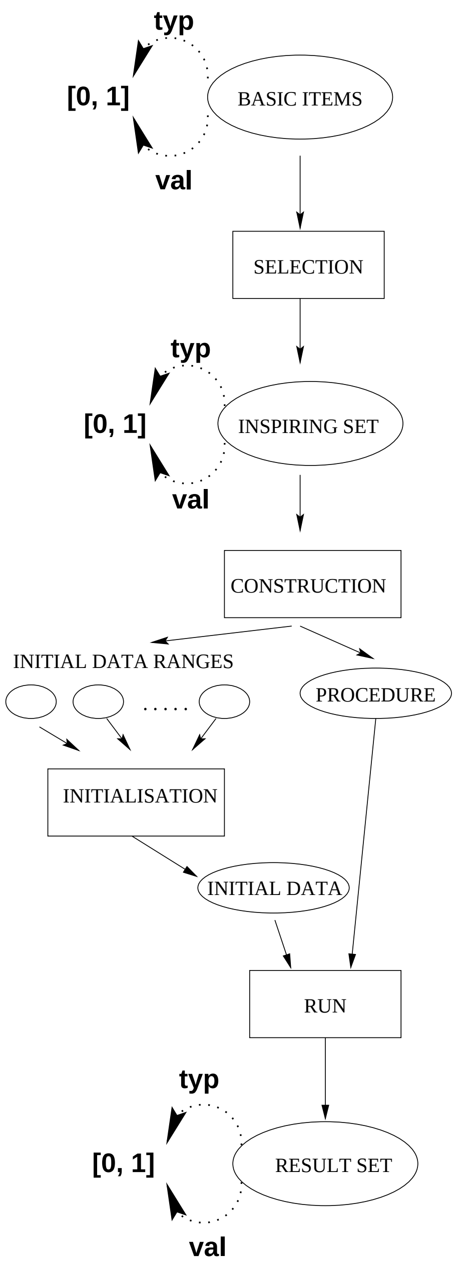

Figure 1: Assumed framework for discussion

such that for $\langle d_1, \ldots, d_k \rangle \in \langle D_1, \ldots, D_k \rangle$, $G(d_1, \ldots, d_k) \in \mathcal{P}(\mathcal{B})$.

In the next two definitions, we assume the concepts and notation of the above definitions.

Definition 7 *An **initialisation** is a mapping $I_{\mathcal{B}}$ from $\mathcal{P}(B) \times \mathcal{RAT}(B) \times \mathcal{RAT}(B)$ to $\langle D_1, \ldots, D_k \rangle$. That is, it is a choice of initial parameters, based on the inspiring set $S_{\mathcal{B}}(typ, val)$ and the ratings schemes typ, val.*

Definition 8 *For a program $(\langle D_1, \ldots, D_k \rangle, G)$, a **run** is a pair $(\langle d_1, \ldots, d_k \rangle, G(d_1, \ldots, d_k))$, where $\langle d_1, \ldots, d_k \rangle \in D_1 \times \ldots D_k$.*

5 Criteria for creativity

The various formal constructs set out above allow us to state some criteria which could be applied in deciding how creative a program is, or has been. We will define these criteria for a single *run* of a program, and assume that the generalisation to a set of runs should be straightforward. We do not consider the idea that the creativity of a program can be considering independently of the sets of results that it produces, i.e. the outputs from its runs. At this early stage, we do not categorically state that all these criteria are essential to an assessment of creativity; rather, they are a first draft of a general catalogue of relevant factors (see also Section 6 below).

In the formal criteria listed below, we assume a value-based artefact class $(\mathcal{B}, typ, val)$, a selection process $S_{\mathcal{B}}$, a construction process $C_{\mathcal{B}}$, an initialisation process $I_{\mathcal{B}}$, a generating program $(\langle D_1, \ldots, D_k \rangle, G)$ for $\mathcal{B}$, and a run of this program $(\langle d_1, \ldots, d_k \rangle, R)$ (where R is finite). It should be possible to generalise these ideas to cover a set or sequence of 'runs', so as to assess the creativity of a program in general rather than a single run of that program. That elaboration is not explored here.

For convenience, we employ the following notation:

$T_{\alpha,\beta}(X) \overset{def}{=} \{x \in X \mid \alpha \leq typ(x) \leq \beta\}$**:** The subset of X falling in a given range of normality.

$V_{\alpha,\beta}(X) \overset{def}{=} \{x \in X \mid \alpha \leq val(x) \leq \beta\}$**:** The subset of X falling in a given range of quality.

$AV(F, X) \overset{def}{=} (\sum_{x \in X} F(x) / \mid X \mid)$**:** The average value of a function F across finite set X.

$ratio(X, Y) \overset{def}{=} \mid X \mid / \mid Y \mid$**:** The relative sizes of two finite sets X, Y.

Consider appraising the output of a generating program in isolation, without knowledge of the program's construction or internal workings. This is comparable to the assessment which people routinely make of human-created artefacts. The extent to which the program produces items which conform to the definition of the genre (i.e. score highly on the typical properties) is relevant (see Section 4.2). The average rating of items should be suitably high:

Criterion 1 *$AV(typ, R) > \theta$, for suitable θ.*

Also, highly rated (i.e. very typical) items should form a significant proportion of the results:

Criterion 2 *$ratio(T_{\alpha,1}(R), R) > \theta$, for suitable α, θ.*

Notice that this in a sense conflicts with the **Novelty** requirement from Section 2 above. As discussed in Section 4.2, merely succeeding in conforming to the norms of the chosen genre is an achievement for a computer program. Branching out into producing untypical items – as suggested by **Novelty** – is a more advanced level of creativity, which we will attempt to capture in further Criteria, below.

Generated items should have intrinsic merit, on average:

Criterion 3 $AV(val, R) > \theta$, *for suitable* θ.

Also, high quality items should make up a significant proportion of the results:

Criterion 4 $ratio(V_{\gamma,1}(R), R) > \theta$, *for suitable* γ, θ.

A program might be successful within the norms of the genre, by having a high proportion of its "normal" output scoring well:

Criterion 5 $ratio(V_{\gamma,1}(R) \cap T_{\alpha,1}(R), T_{\alpha,1}(R)) > \theta$, *for suitable* α, γ, θ.

As Boden (1992) makes clear, a higher rating of creativity should be accorded to the production of artefacts which do not conform closely to the norms of the genre, but which nevertheless are rated highly when judged on their merits. We can model this judgement by comparing the set of untypical high-valued items with either the entire set of outputs, the whole set of untypical items, or the set of typical high-valued items. Each of these ratios could constitute a criterion of creativity:

Criterion 6 $ratio(V_{\gamma,1}(R) \cap T_{0,\beta}(R), R) > \theta$, *for suitable* β, γ, θ.

Criterion 7 $ratio(V_{\gamma,1}(R) \cap T_{0,\beta}(R), T_{0,\beta}(R)) > \theta$, *for suitable* β, γ, θ.

Criterion 8 $ratio(V_{\gamma,1}(R) \cap T_{0,\beta}(R), V_{\gamma,1}(R) \cap T_{\alpha,1}(R)) > \theta$, *for suitable* $\alpha, \beta, \gamma, \theta$.

A creative program might well replicate most of the inspiring set $S_{\mathcal{B}}(typ, val)$:

Criterion 9 $ratio(S_{\mathcal{B}}(typ, val) \cap R, S_{\mathcal{B}}(typ, val) > \theta$ *for suitable* θ.

Producing more than just the inspiring set is a symptom of creativity:

Criterion 10 $ratio(R, S_{\mathcal{B}}(typ, val) \cap R) > \theta$, *for suitable* θ.

Results which are not in the inspiring set should be at least typical of the genre, and better still highly-valued:

Criterion 11 $AV(typ, (R - S_{\mathcal{B}}(typ, val))) > \theta$, *for suitable* θ.

Criterion 12 $AV(val, (R - S_{\mathcal{B}}(typ, val))) > \theta$, *for suitable* θ.

Also, well rated items not in the inspiring set should be a significant proportion of the results:

Criterion 13 $ratio(T_{\alpha,1}(R - S_{\mathcal{B}}(typ, val)), R) > \theta$, *for suitable* α, θ.

Criterion 14 $ratio(V_{\gamma,1}(R - S_{\mathcal{B}}(typ, val)), R) > \theta$, *for suitable* γ, θ.

The relationship of the inspiring set to the *program* is also of interest. This is less easy to formalise, because the relevant issue is the extent to which human intervention – the normal mechanism – is used. That is, each of the $S_{\mathcal{B}}$, $C_{\mathcal{B}}$ and $I_{\mathcal{B}}$ mappings could be human-crafted or automated; $S_{\mathcal{B}}$, and $I_{\mathcal{B}}$ could be random. The creativity of the program could be assessed according to these aspects, with human-crafting (probably) leading to a lower evaluation.

6 Viewpoints

There are (at least) two respects in which subjectivity or relativistic judgement enters into the assessment of creativity.

Firstly, the appraisal of the program's output, modelled here by the class-rating and the value-rating, are, particularly in artistic fields, highly personal. This can be modelled by having the two rating schemes (typ, val) act as a formalisation of the judge's viewpoint. That is, any formal (or semi-formal) definitions of the quality of the program results can be stated to be relative to the rating schemes involved, and hence to the subjective view of a particular judge.

Secondly, there is no consensus on what counts as creative, particularly when considering programs. A framework such as the one outlined here allows for multiple definitions of creativity (or definitions of different styles or levels of creativity). As mentioned in Sections 4.2 and 5, for a computer to manage even to produce "normal" or "typical" exemplars of a genre (thus scoring well on Criteria 1 and 2) is a worthwhile task, but it is a different level of achievement from producing highly-valued but untypical artefacts (Criteria 6, 7, 8). This emphasises that the set of Criteria listed here should be considered as a repertoire from which one might draw. The fact that different Criteria seem to lead in different directions with respect to the underlying intuition about creativity is not a problem. Rather, one can define different variants by suitable choices and combinations of criteria.

More formally, the various parameters $\alpha, \beta, \gamma, \theta$ in the above definitions are a source of flexibility. Also, whatever criteria are formally defined (those given above being an illustrative set) can be put into different logical combinations, and various weights could be attached to them. We could have what was described earlier as a *weighted rating scheme*, but for judging the creativity of a program (based on its origins and its output), rather than for assessing the characteristics of the output.

If we could settle on a set of criteria such as those listed in Section 5 above, then we could formally define a *creativity judgement system* as being a set of values for the various parameters involved $(\alpha, \beta, \ldots)$. However, it is premature to frame such a definition. We need to refine our ideas about suitable criteria (and suitable parameters) before attempting standardisation in this way.

7 Practicalities

The ideas presented here are metatheoretical, and are intended to contribute to a methodological and philosophical debate. The natural question is: can any of this be applied, in practice, to real programs? The answer is that the full range of devices outlined in Section 5 above is unlikely to be suitable for detailed assessment of all generating programs. Some of the criteria are more tractable than others. In many cases, it should be feasible to compute precise values for Criteria 1, 2, 3, 4, 5, 7, 8 for suitable values of the parameters $(\alpha, \beta, \gamma, \theta)$. For example, the output of the JAPE joke-generator was evaluated by human judges against two standards: 'is this item a joke?' and 'how funny is this item?' (Binsted et al., 1997). These correspond directly to the typical rating and the value rating of our framework, so (given the complete raw data from that evaluation) these Criteria could be evaluated.

However, the way in which many programs are developed means that the 'inspiring set' is usually not recorded or documented precisely, but typically forms part of the informal, unstructured thinking that guides the design process. Hence Criteria 9, 10, 11, 12, 13, 14 will often be very hard to measure. These criteria merit inclusion in our list nevertheless, as they make explicit and precise some important standards which are often applied, albeit informally. Also, there will be cases where they can be computed with great precision: when the construction process $C_{\mathcal{B}}$ is carried out by machine learning techniques of some sort, the training set used could reasonably function as the inspiring set.

8 Boden's classification

Boden (1998) presents a classification of styles of creativity as manifested by computer programs, deriving from her 'conceptual space' perspective (Section 3 above). (The current paper began as an attempt to formalise this classification)

From a consideration of human and computational creativity, she offers three categories of computer creativity: *combinational* ('novel (improbable) combinations of familiar ideas'), *exploratory* ('generation of novel ideas by the exploration of structured conceptual spaces'), and *transformational* ('transformation of some (one or more) dimension of the space so that new structures can be generated which could not have arisen before').

Owing to the lack of formality and detail in Boden's presentation, it is not entirely clear how these forms are distinguished. In particular, the distinction between *combinational* and *exploratory* is hard to pin down. Boden cites the JAPE program (Binsted, 1996), which creates riddles according to the various sets of rules which Binsted programmed into it. In a sense, JAPE explores a space of possibilities in a structured fashion, which seems to be *exploratory*, but Boden offers it as an example of *combinational* creativity, and contrasts this with a system (Koning and J.Eizenberg, 1981) for generating possible house designs (within a particular genre) according to a shape-grammar, which she does regard as *exploratory*.

Boden uses the term 'conceptual spaces' when talking of 'exploration' or 'transformation', and says that a program's search space is a special case of a conceptual space. Whether considering abstract conceptual space or implemented search space, it is hard to see the evidence for her claim that the AM program (Lenat, 1979) *transforms* the space whereas the house-design program merely *explores* it.

In terms of our formal constructs, exploratory creativity might be roughly characterised as a process in which: (a) most of the items produced are within the norms of the genre; (b) all the valuable items it produces are within the norms of the genre. In the formal terms used above, (a) would be Criterion 2, and (b) might be:

$V_{\gamma,1}(R) \subseteq T_{\alpha,1}(R)$ for suitable α, γ.

Transformational creativity seems to be a situation in which the program produces a significant set of items with a high value-rating and a low class-rating (Criterion 6, above), but which would have a high class-rating according to some (yet to be defined) rating scheme typ' (with associated $T^{'}_{x,y}$):

$$V_{\gamma,1}(R) \cap T_{0,\beta}(R) \subseteq T^{'}_{\alpha,1}(R)$$

for suitable α, β, γ and $T^{'}_{x,y}$.

9 Possible extensions

The ideas presented here are rather tentative and preliminary. There are a number of ways in which they could be elaborated to capture more subtle aspects of creative processing.

9.1 Program construction

It would be interesting to develop further the issue mentioned at the end of Section 5 above: how does the program come into being, and what is its exact relationship to the inspiring set?

9.2 Self-rating of output

Boden (1992, p.83) suggests that one facet of human creativity is an ability to recognise the worth of a created entity. At a very crude level, we could allow for a generating program assigning some sort of rating to each item that it produces, indicating the value that the program allocates to that item. Some programs (e.g. AM (Lenat, 1979)) include a mechanism of this sort. It should be relatively straightforward to modify the formal definitions given in Section 5 above to accommodate this form of data. The amended criteria should then capture the intuitive idea that the program's rating of its own output ought (if the program is to be deemed creative) to have a high correlation with either the typical ratings or the value ratings or both.

9.3 Computations and traces

As noted above, Boden deems a program to be creative only if it operates (internally) in a certain way. Colton (personal communication) has suggested that if a program run produces a set of output items which is highly-rated (by typical and/or value ratings) in a relatively simple manner, it should be regarded as more creative than a comparable program run which is more laborious in producing the same results. Although the framework here concentrates on the external aspects of a program, and deliberately leaves open the question of what computations count as inherently creative, it is conceivable that the "elegance" or "simplicity" of the program's deliberations could be taken into account as one more factor in the assessment of creativity. We would have to make changes to the formal definitions given in Section 4.5 above. In outline, we could do it in the following way (replacing Definitions 5 above):

Definition 9 *Given a set of basic items $\mathcal{B}$, a* ***generating program*** *for $\mathcal{B}$ consists of a pair $(\langle D_1, \ldots, D_k \rangle, G)$ where $\langle D_1, \ldots, D_k \rangle$ is a k-tuple of sets, and G is a mapping from $D_1 \times \ldots \times D_k$ to a pair (t, S), where $S \in \mathcal{P}(\mathcal{B})$ and t is a* ***trace****.*

Definition 8 could remain unaltered, but a 'run' would now be a pair whose second component is itself a pair, consisting of a trace and a result set. The result set would continue to figure in the Criteria in the same way, indicated by R as before.

Informally, a "trace" here is a representation of the steps taken by the program. To make this fully rigorous, we would need clear definitions of notions such as "possible set of operations" by a program and "instances of operations". A "trace" could be a sequence of instances of operations, and a program would characterise a (possibly infinite) set of possible traces.

The assessment of creativity would then depend not only, as before, on two rating schemes for output items, but would also include a *simplicity metric* which was defined to map 'trace' structures to the interval [0,1].

This would admit, in a relatively limited manner, one further factor (simplicity of derivation) into our way of appraising creativity. It would *not* concede Boden's (apparent) claim that we can make the *manner* in which the generating program operates a *necessary* condition for a verdict of "creative".

9.4 Random generation

Since the formalisation in Section 4 above says little about *how* artefacts are generated, there is a sense in which it does not exclude the random production of basic items. However, the level of abstraction of the framework means that it has no way to distinguish random generation from any other approach. Although random generation is not a hugely interesting case from an AI point of view, it might, in certain discussions, be an interesting yardstick for comparison purposes. To allow explicit formal comparisons, we would need a more detailed definition of the available space of basic items and how this space could be randomly explored. One way to do that might be to decompose the notion of 'basic item' so as to make explicit the internal structure of an item (e.g. as an array of pixels, or as a sequence of words). The other requirement would be a suitable definition of "random combination of atomic parts".

9.5 User guidance

The framework does allow at least a rough characterisation of the situation where a program is "tuned" to produce certain data. That would be defined as a case where the inspiring set and the result set are identical, or almost so. Such a scenario might be rated poorly in creative terms (it would score badly on Criteria 10, and, *a fortiori*, on Criteria 13 and 14), but it could still be interesting from a research point of view, since it might well demonstrate, and test by implementation, a mechanistic route from very simple data to interestingly complex output. (As remarked by Ritchie and Hanna (1984), even if Lenat's AM had been designed specifically to compute particular mathematical concepts from a small set of primitives, that would be interesting, even if it might be less highly rated for creativity.)

What is not describable in our formalisation is the situation where a user intervenes during the running of the program, using knowledge of what the program has already done in order to guide it in particular directions. To make explicit statements about this kind of activity, the process of computation would have to made explicit (cf. comments in Section 9.3 above).

9.6 Similarity

An important weakness of the framework is that it does not handle the notion of *similarity* very cleanly. The **Novelty** standard of Section 2 is covered only indirectly and partially. It would be interesting to develop the idea mentioned in Section 4.3 above, by building some notion of multi-dimensional space into the assessment framework. Not only might this connect to some of Boden's ideas, it would allow the notion of similarity to be addressed more directly. In particular, the present set of Criteria compare the result set with the inspiring set only in terms of overlap in membership, but make no allowance for the (probably pertinent) idea that output items could vary in the extent to which they are similar to those in the inspiring set.

10 Conclusion

We have proposed an approach to the assessment of creativity (in programs) which:

(a) leaves open the question of which mechanisms might lead to creativity;

(b) provides a general sketch of what might constitute a creative program;

(c) highlights various factors which are relevant to an empirical judgement of creativity (about a program);

(d) is explicit and formal about these factors;

(e) allows for "subjectivity", i.e. variations in the criteria used to judge the output items;

(f) allows for "relativity", i.e. variations in the criteria used to define creativity.

These proposals are very preliminary; Section 9 lists just some of the ways in which they could be improved. We hope that this framework will clarify the discussion, and may lead to a sounder basis for attributing creativity to programs.

Acknowledgements

I would like to thank Geraint Wiggins and Simon Colton for discussions of some of these ideas.

References

Kim Binsted. *Machine humour: An implemented model of puns*. PhD thesis, University of Edinburgh, Edinburgh, Scotland, October 1996.

Kim Binsted, Helen Pain, and Graeme Ritchie. Children's evaluation of computer-generated punning riddles. *Pragmatics and Cognition*, 5(2):309–358, 1997.

Margaret A. Boden. *The Creative Mind*. Abacus, London, 1992.

Margaret A. Boden. Creativity and Artificial Intelligence. *Artificial Intelligence*, 103:347–356, 1998.

Lynne Cahill, Christy Doran, Roger Evans, Daniel Paiva Chris Mellish, Mike Reape, Donia Scott, and Neil Tipper. In search of a reference architecture for nlg systems. In *European Workshop on Natural Language Generation*, Toulouse, 1999.

Simon Colton, Alan Bundy, and Toby Walsh. Agent based cooperative theory formation in pure mathematics. In G. Wiggins, editor, *Proceedings of AISB 2000 Symposium on Creative and Cultural Aspects and Applications of AI and Cognitive Science*, pages 11–18, Birmingham, UK, April 2000.

H. Koning and J.Eizenberg. The language of the prairie: Frank Lloyd Wright's prairie houses. *Environment and Planning B*, 8:295–323, 1981.

D.B. Lenat. On automated scientific theory formation: a case study using the AM program. In Hayes, Michie, and Mikulich, editors, *Machine Intelligence 9*, pages 251–283. Ellis Horwood, Chichester, 1979.

G. D. Ritchie and F. K. Hanna. AM : A case study in ai methodology. *Artificial Intelligence*, 23:249–268, 1984.

Graham Steel, Simon Colton, Alan Bundy, and Toby Walsh. Cross-domain mathematical concept formation. In G. Wiggins, editor, *Proceedings of AISB 2000 Symposium on Creative and Cultural Aspects and Applications of AI and Cognitive Science*, pages 3–10, Birmingham, UK, April 2000.