Creativity versus the Perception of Creativity in Computational Systems

1 A Position Statement

In certain domains, the value of a created artefact can be assessed with little regard to the process which created it. For instance, in linguistics, there is no need to consider how a joke was produced in order to appreciate its humour. The same is true in scientific discovery, where it is often possible to objectively measure the value of a concept or hypothesis independently of the process which generated it. In other domains, however, the value of a created artefact is best characterised as a function of many aspects, some of which will relate to the artefact itself and others to the production process. Applied to computational creativity, this means that, in order to project the word 'creative' onto a piece of software, people need to understand – at least at a high level – how the program works, and be able to compare this to how humans create.

In the visual arts, in particular, the subjective value of a painting will depend on: how well it adheres to the aesthetics of the viewer, how personalised it is to that viewer, the perceived effort which went into producing it, the perceived level of experience and – importantly – the perceived creativity of the artist. This has been borne out most strikingly in modern artistic movements, many of which almost completely ignore the aesthetic value of the artwork when evaluating it, concentrating more on the innovations in the production process and in art theory which underly the artwork. In extreme cases – for instance, in some of the conceptual art works of Duchamp – an artwork exists entirely as testament to the creative/intellectual act rather than as an artefact to be treasured in its own right. Even in more traditional circles, it is clear that the creative process at work (or perceived lack of it) affects the value that people assign to paintings. This is most obvious in computer arts: to a particular viewer, the visual appeal of a simulated watercolour produced in seconds by Photoshop may match the visual appeal of a painting on a canvas produced by a trained artist over a series of days. However, the overall appeal will probably be much less for the computer generated artwork, because there will be a perception of the artist using his/her creativity and experience in the production of the painting, and this will not be similarly projected onto the computational process.

This has led to a vicious circle where computers may produce aesthetically pleasing artwork, but because they are initially perceived as being uncreative (for various reasons), their art will not be evaluated highly, and without highly valued art, they will never be judged as being creative, which compounds the problem. Fortunately, the situation for computational creativity in the visual arts is not entirely irredeemable. To begin to rectify matters, we need to realise that, especially for software, there is a difference between being creative and being perceived as creative by art consumers and critics. In order for software to be perceived as creative, it must exhibit behaviours which can be described as (a) skillful (b) appreciative and (c) imaginative. Once we can project these behaviours onto our software, if the artefacts produced by it are evaluated as being valuable, we should be able to call our software creative.

2 Overview of the Full Paper

Due to space restrictions, it was necessary to present the above situation assessment as a position statement, with little argumentation or evidence for some of the points made. Naturally, in the full paper, we rectify this. In particular, we draw upon art theory and philosophical texts to justify our position that process is equally as valid as artefact quality in the assessment of creativity in general, and computational creativity in particular. We also firmly place our approach to the assessment of creativity in software into context with respect to the existing work in this area. In particular, in an ideal world (or at least, ideal for computational creativity research), we would be able to judge creativity using only the artefacts produced, so that there can be no prejudice towards computationally creative systems. This ideal exists in certain application domains, and so schemes for assessing creative software in terms of its output are valid. A particularly useful and comprehensive set of such evaluation criteria is given in [2]. However, as argued above, we often do not have a level playing field. Therefore, in response to [2], in the full paper, we suggest augmentations of existing schemes for the assessment of creativity in software which take into account evaluations of how the software may be perceived. To do this, we introduce the notion of a *creative tripod* (of skill, appreciation and imagination) which supports creative behaviour.

We then perform a survey of existing software which has historically been described as creative, and discuss – according to the projection of skill, appreciation and imagination onto a system's processes – whether or not certain software is likely to be perceived as creative. As a good example, Cohen's AARON program has some imagination and skill in generating and then painting scenes containing people and plants in rooms. However, to the best of our knowledge (and after conversations with Cohen), it seems unlikely that any of AARON's processes have an appreciation of what they are doing – although this situation could probably be rectified, for example by incorporating machine learning software. Hence, AARON is open to the criticism that it does not appreciate its subject matter, or its tools, hence cannot really be creative. We use the HR automated mathematical theory formation system [1] and The Painting Fool visual arts software (`www.thepaintingfool.com`) as case studies. We developed the HR system to be creative with criteria such as those from [2] in mind, while we developed The Painting Fool to be perceived as creative in the way it operates. In the full paper, we discuss how this subtle but important difference might influence the writing of creative systems in general.

References

1. S Colton. *Automated Theory Formation in Pure Mathematics*. Springer-Verlag, 2002.
2. G Ritchie. Some empirical criteria for attributing creativity to a computer program. *Minds and Machines*, 17:67–99, 2007.