

Measuring the Complexity of All the Art

Jonathan Langke and Peter Boothe

August 4, 2013

Abstract

We define formula complexity based on a restriction of formula complexity, and measure the formula complexity of all possible 2×2 and 3×3 artworks. We have also conducted a survey of the relative perceived visual complexity of these artworks, and we report that survey's results. We end by discussing the links between these two notions, and find that formula complexity does seem to be related to visual complexity.

1 Introduction

We explore the complexity of simple digital artworks, both from a formal mathematical standpoint and from a human experience standpoint. Art has many definitions, but the one we will use is that a piece of black and white digital art is a matrix where every entry is either true or false. This can be thought of as a matrix of boolean values, where true corresponds to black and false corresponds to white, or it can be thought of as the output of a formula. An artwork can be generated from a mathematical formula that evaluates to a boolean and involves integers x and y . For our initial study, we consider only 2×2 matrices, because there are only $2^{2 \cdot 2} = 16$ of them, which allows for careful study of each step in our analysis pipeline. Once we see the properties of 2×2 artworks, we extend our work to 3×3 artworks.

The formula complexity or program-size complexity of a particular object is the size of the smallest program which outputs that object. This allows us to talk about the complexity (and randomness) of individual items. Formula complexity is, unfortunately, uncomputable. It is, however, recursively enumerable (RE). Therefore, we can enumerate all well-typed expressions and evaluate them on multiple inputs. The size of the smallest expression which outputs that object is then its formula complexity!

Our algorithm to generate all the art, and keep track of its formula complexity is as follows: Enumerate all well-typed formulae of size 1. For each enumerated formula, generate its corresponding artwork. If this artwork has not been previously generated, then it has Kolmogorov complexity 1. Next, we repeat for size 2, 3, 4, etc., until we have managed to generate all possible 2×2 artworks. To do this, however, we must first define what we mean by a “well-typed formula”.

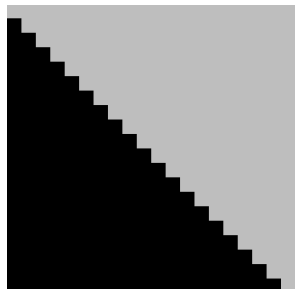
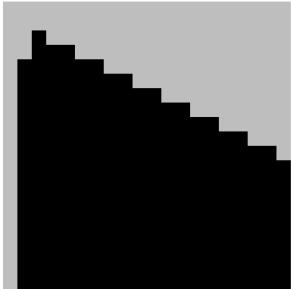
Expression	$(x < y)$	$((1 + (1 + 1)) < (x * y)) \text{ and } ((x + 1) < (y * (1 + 1)))$
Expression Size	3 symbols	19 symbols
Artwork		
Formula Complexity	3	19

Figure 1: Two artworks of differing formula complexity

A formula is well-typed if the inputs types of all functions¹ match the type of those function’s arguments. An example of an ill-typed formula is `(+ false 3)`, whereas a well-typed formula is `(+ 3 x)`. We further require that the overall type of our formula be `bool`. An example of such an expression is `(< x y)`, which is an expression involving integers with two inputs that evaluates to either true or false.

Further, we also need to define our programming language with which to express our formulae. Our language is constructed out of the atoms `< + * x y 1 0 and or not true false`. As a side note, every formula in our language is also a valid Racket program[1], which greatly aided us in evaluation.

2 Formula Complexity

Formula complexity is a “low-powered” version of Kolmogorov complexity². Kolmogorov complexity allows the full power of a programming language to be put to bear in describing how an output is produced. In particular, Kolmogorov complexity allows for the definition and application of functions as well as conditional statements. In our language, the only things which can be used are formulae with a boolean result involving only the logical atoms `and`, `or`, `not`, `true`, and `false`, as well as the numerical atoms of 0, 1, `x`, `y`, `+`, `*`, and `<`.

Because formula complexity disallows loops and function definitions, every formula represents a terminating program. Even more, since the domain of all of the operators is

¹So as not to get distracted by issues of operator precedence, formulae will often be presented fully parenthesized in prefix notation.

²The standard text for Kolmogorov complexity is Li and Vitanyi[4]

total, this means that every well-typed formula represents a non-crashing program³. Thus, in exchange for “powering down” Kolmogorov complexity, we can actually enumerate and test all formulae, without worrying about infinite loops or crashes.

Each formula evaluates to a boolean and involves the variables x and/or y . Therefore, to create a 3×3 artwork from a formula, simply evaluate the formula at all of the points $(x, y) = (0, 0)$, $(x, y) = (0, 1)$, $(x, y) = (0, 2)$, $(x, y) = (1, 0)$, \dots , and $(x, y) = (2, 2)$. Where the formula evaluates to true, set the corresponding square on the art to black; where the formula is false, set the square to white. Every artwork has multiple formula which produce it. The formula complexity of an individual artwork is the smallest such formula.

As an example, the all-black is produced by the formulae **true**, **not false**, **true or false**, and many many others. However, because the smallest such formula consists of just one atom, the formula complexity of the all-black artwork is 1. From this example (and from the definition), we see that formula complexity is a property of an artwork, not a formula. The formula complexity is defined as the size of the smallest formula able to produce the output in question.

2.1 Generating The Art

To generate all the art, we first generate all well-typed boolean formulae in order of size. Then, for each formula, we generate its corresponding artwork. If this is the first time that artwork has been generated, we then know that the formula complexity of the artwork is equal to the size of the formula just used. Generating all the 2×2 formulae revealed some hidden structure to our results.

2.1.1 All the 2×2 Art

From Figure 2, some things are immediately obvious. First of all, as might match one’s intuition, the all-black and all-white pictures are the least complex pictures. Secondly, one can see that if a formula of size n exists to create a certain picture, then that picture’s inverse has a complexity of one of $n - 1$, n , or $n + 1$. Examples of this last phenomenon can be seen between levels 7 and 8, as well as between levels 3 and 4. This occurs for the simple reason that adding one symbol (**not**) creates a picture’s inverse. Also, somewhat intriguingly, we can see gaps at 2 and 6. Although there are formulae of size 2, none of those formulae produce a picture that has not been already produced by formulae of a smaller size e.g (**not true**) and (**not false**) are more succinctly expressed as **false** and **true**.

Also visible in our results is that if an artwork has formula complexity of n , then the diagonal flip of that artwork also has formula complexity of n , because exchanging x and

³This is why the division and modulus operators were excluded, as $12 \bmod 0$ and $12 / 0$ result in a crashing program rather than a value.

Formulae	Level	Pictures
none	0	empty
(true), (false)	1	
none	2	empty
$(< x\ 1), (< y\ 1), (< x\ y), (< 0\ x), (< 0\ y), (< y\ x)$	3	
$(\text{not } (< x\ y)), (\text{not } (< y\ x))$	4	
$(< (y + x)\ 1), (< (y * x)\ 1), (< 0\ (y + x)), (< 1\ (y + x))$	5	
none	6	empty
$(\text{or } (< y\ x) (< x\ y))$	7	
$(\text{not } (\text{or } (< y\ x) (< x\ y)))$	8	

Figure 2: All the 2×2 art with its corresponding complexity.

y results in a formula of exactly the same size. This can be seen in levels 3, 4, and 5 of Figure 2.

After generating all 2×2 artworks, we found that the artworks were too small for people to feel comfortable discussing those artworks’ visual complexity. Therefore, we set our sights higher.

2.1.2 All the 3×3 Art

Generating all of the 3×3 art proved to be a more significant computational challenge. Even with some rudimentary pruning, we still had tens of billions of formulae to check. Our preliminary results may be seen in Figure 3. Note that although there were no new artworks in level 6 in the 2×2 results, there are new artworks at that level in the 3×3 artworks — these new artworks differ from smaller ones only in their last column or row, and so these differences were not relevant to the 2×2 case.

After generating all the 3×3 artworks, we wished to discover whether our notion of formula complexity had anything to do with visual complexity.

3 Assessing Visual Complexity

There is an intuitive notion of “visual complexity”, but it is unclear whether any two people would agree on whether a particular artwork is visually complex. It is also not known whether visual complexity is related to computational complexity. In an effort to discover whether visual complexity is something that people agree upon, and to discover whether

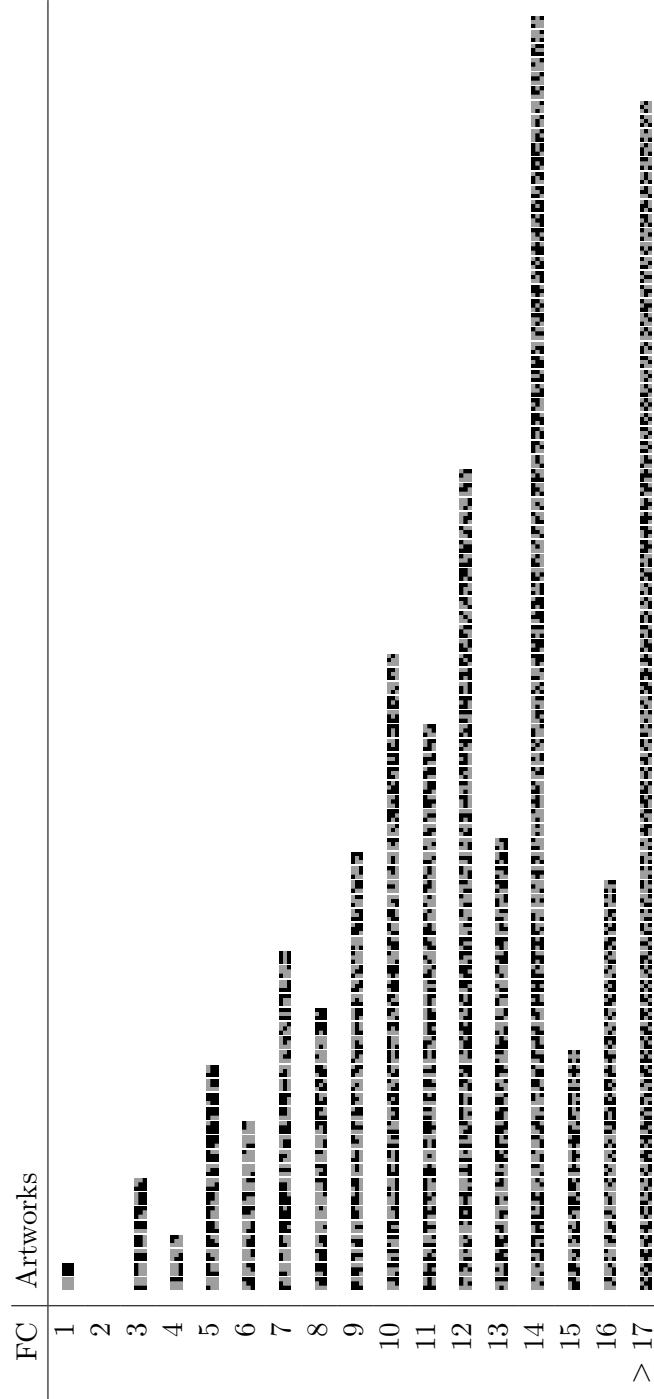


Figure 3: All the 3×3 art with its corresponding complexity.

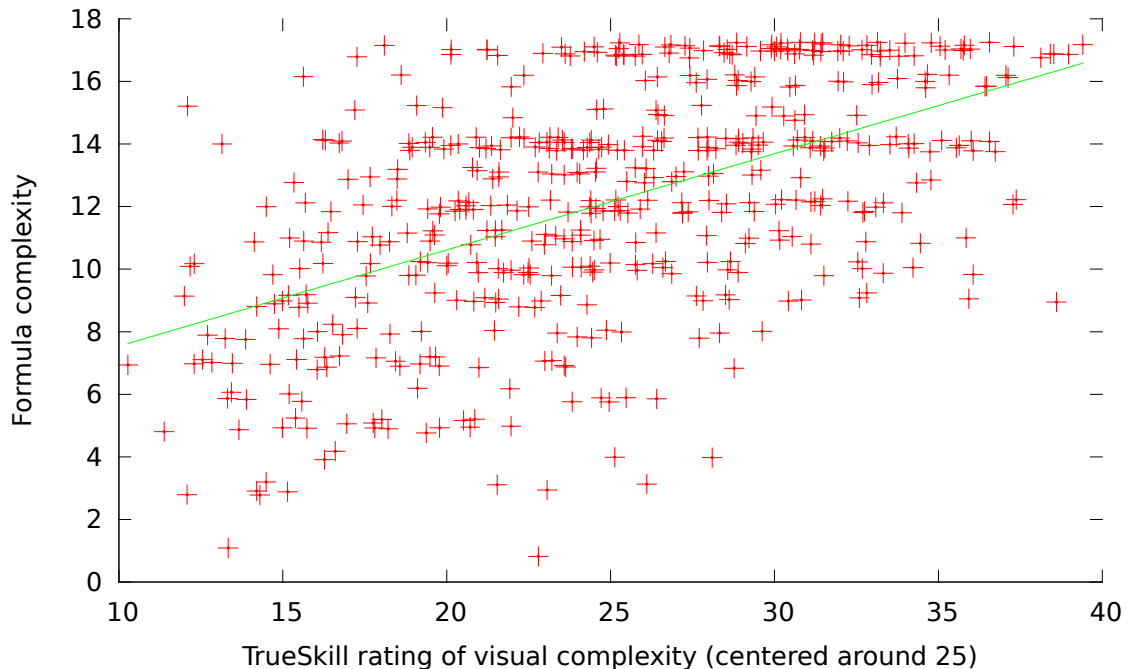


Figure 4: Formula complexity vs visual complexity (all points jittered by up to .25) with line of best fit. Formula complexity has been clamped at 17. Each data point represents one of the 512 possible 3×3 artworks.

formula complexity is related to perceived visual complexity, we conducted a survey via a website.

We constructed two websites that asked visitors to choose which of two images they found to be of greater visual complexity. One website asked about all the 2×2 art, and the other asked about all of the 3×3 art. Both such sizes are small enough to completely enumerate (there are 16 and 512 artworks in each respective category) as well as small enough to calculate the formula complexity of each artwork.

We then asked users on Twitter as well as local campus students to repeatedly assess which of two images they found more visually complex. We treated each assessment as if it were a competition between two players (artwork one vs artwork two), and when the user decided upon a winner, we used the TrueSkillTM[3] algorithm to update each artwork’s relative visual complexity based on each match’s results.

4 Combined Results

Once we have a measured formula complexity (Section 2) and a measured visual complexity (Section 3), we plot them together. The resulting plot may be seen in Figure 4. Also included in that plot is a line of best fit. Visually examining the plot, we see that there might be correlation. Actually computing the Pearson correlation coefficient results in a correlation coefficient of 0.53 and an extremely low p-value ($4 \cdot 10^{-39}$). All of which implies that the two measurements are strongly correlated and it is very unlikely that this result is due to chance. Therefore, formula complexity is correlated with visual complexity!

5 Summary and Future Work

We defined a notion of formula complexity, which served as a “low power” version of Kolmogorov complexity. We enumerated all well-typed formulae to discover the formula complexity of a large set of very small artworks. We then asked a self-selected set of people to assess the visual complexity of the artworks. We found a nice relationship between perceived visual complexity and formula complexity, although our results may suffer from the strong law of small numbers[2]. A relationship between visual complexity and formula complexity suggests that how our brains decide whether something is “complex” may have a lot to do with whether that item is computationally complex! We found this to be quite an exciting result, as it draws a direct connection between the notion of complexity in two separate domains: brains and computational machines.

Anybody interested in reproducing or extending our results is encouraged to fork our repository on GitHub⁴ and use the code however you see fit. Our data and analysis pipeline are all in the repository as well. Extending our results to larger artworks would, in particular, be quite interesting. Discovering what languages have formula complexity that best matches the perceived visual complexity would also be quite interesting, as it would help explicate this deep link between complexity of perception and complexity of computation.

References

- [1] Daniel P. Friedman and Matthias Felleisen. *The Little Schemer*. MIT Press, 1995.
- [2] Richard K. Guy. The Strong Law of Small Numbers. *The American Mathematical Monthly*, 95:697–712, 1988.
- [3] Ralf Herbrich, Tom Minka, and Thore Graepel. TrueSkill(TM): A Bayesian Skill Rating System. In *Advances in Neural Information Processing Systems 20*, January 2007.

⁴<https://github.com/JLangke/KComplexity>

- [4] Ming Li and Paul M. B. Vitányi. *An introduction to Kolmogorov complexity and its applications (2. ed.)*. Graduate texts in computer science. Springer, 1997.