A proposal for a bachelor thesis based on a "measure" of "complexity"

Henrik Sommerland (henrik.sommerland@gmail.com)

March 11, 2014

Abstract

In this text I will give a short introduction to a method i have devised for measuring the complexity of strings and language. The purpose of this text is to get feedback on weather or not this is suitable for a bachelor thesis. Any form of feedback is greatly appreciated. Especially pointers to related or equivalent work

1 The method

1.1 Measuring the complexity of a string

The method for measuring the complexity of the string is very straight forward. It is simply put just the number of unique symbols in the run length encoding of the string. Example:

$$w = aaaabbabbaababbaabbb$$

$$rle(w) = a^4b^2a^1b^2a^2b^1a^1b^2a^2b^3$$

$$c(w) = 6$$

I will from here on write the complexity of a string w as $c_m(w)$ where m is the number of different symbols in the string.

1.2 Some key formulas

1.2.1 The length required for a specific complexity

One question which might arise is what is the minimum length required to have a maximum complexity c where the string can contain m different symbols. This is quite easily obtained trough the formula :

$$l_m(c) = \sum_{i=0}^{c} \left\lfloor \frac{i}{m} \right\rfloor + 1 \tag{1}$$

1.2.2 The maximum complexity for a given length

No on to a slightly more tricky problem. We now want a way to compute the maximum possible complexity for strings of a give length l. This essentially means that we have to invert (1). There exists no formula on closed form which gives the inverse of (1). Although I have found a neat way of finding a nice approximative solution which holds up very well when $l >> m^2$. Using that approximation the maximum complexity becomes:

$$cmax_m(l) = \sqrt{2ml + \frac{m^2}{4}} - \frac{m}{2} \tag{2}$$

2 Results so far

As of now most of my results are totally experiment based. I have had some theoretical which even though not numerous they are still somewhat promising.

2.1 Elementary cellular automata

I have applied my method to the rulespace of the 1D elementary deterministic cellular automatons.I found that my method picked out Rule 110 (and its equivalent rules) as the rule with the highest complexity of all the 256 different rules. Now this is interesting since Rule 110 is capable of universal computation.

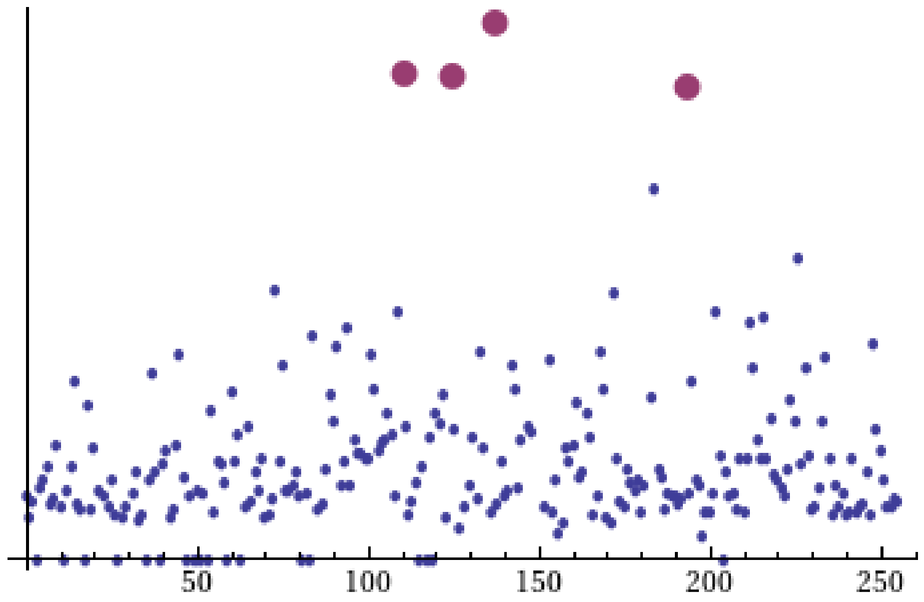

Figure 1: A plot over the complexities of each of the 256 elementary CA. The thicker point is rule 110 and its equivalent rule. Note that the actual values is irrelevant and I only look at the relation to the values of the other rules.

I generalized my method of complexity measures to the automatons by measuring the complexity of each cell as a function of time and then combining the results trough taking the complexity again of the complexities of the list of these new complexities and multiplying it by the highest complexity in the list. Then doing this several times for different initial configurations and averaging the results.

I have repeated this result using some variations of the methods of combining the complexities of the cell histories.

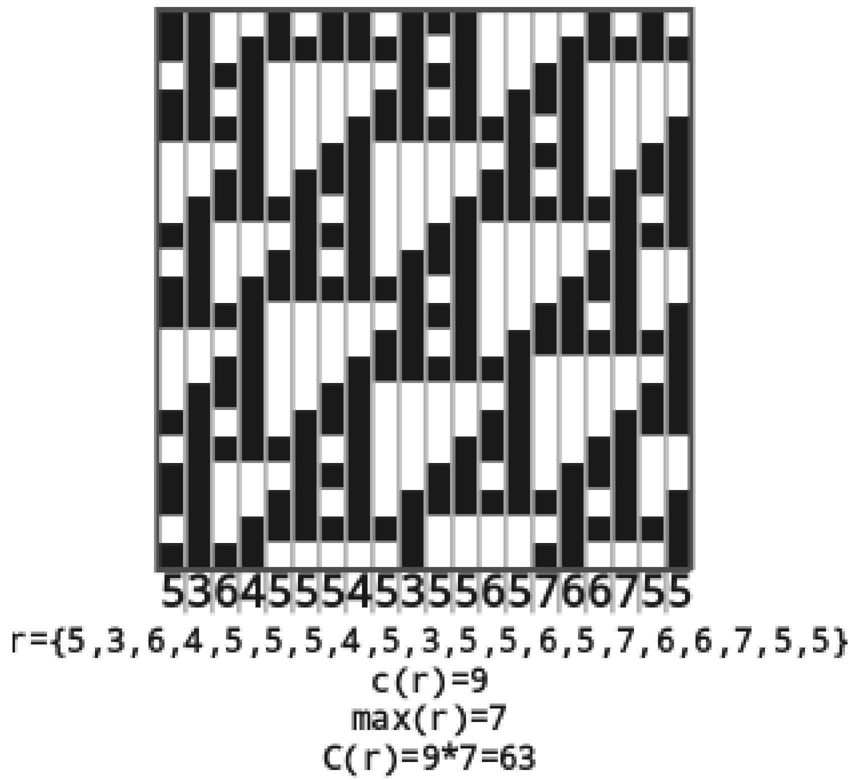

Figure 2: An illustration of the method used to get the complexity of the automaton C(r)

2.2 2D non deterministic cellular automatons

I also applied my method to the rule space of the 2D nondeterministic 9-neighborhood cellular automatons in a similar fashion as with the 1D elementary CAs. I have only run this on roughly 50000 out of the 2^{18} rules but out of those 50000 rules it picked out the rule for conways game of life. Allthough when I searched trough the entire rulespace iw as not able to get CGOL to become the top rule. CGOL ended up in the top 100 though. I will have to redo the search trough the entire rulespace with more samples and larger boards. Now this is quite neat since game of life is also capable of universal computation. More interesting still is that the rule 110 and game of life are radically different i n the way a universal Turing machine is built with them. The proof of universality of rule 110 is based on a cyclic tag system where as game of life has a far more flexibility when it comes to constructing post-Turing machines.

2.3 Work to be done

Well there is definitely a lot of work left to be done. Apart from figuring out a effective way of computing the average complexity of a random string. I must also repeat all of my results using more stringent methods and proper provable code. There is also a need to test several different systems apart from the one described here to try to reproduce the results.

It would also be nice to figure out a way of normalising the complexity of languages. Since the scale is completely arbitrary right now I can not make comparison of the complexity between two different systems (such as rule 110 and game of life).

I have also done some preliminary tests for using this system for finding signals on a very noisy background.

But the big thing still remains. I have as of now only a vauge idea why this works as well as it appears to do. Also if I continue to find more examples where my method manages to pick out machines capable which are Turing complete. It would be very nice to be able to give good arguments for why this is.