Random Bullshit

Christopher Abad
20 GOTO 10 — aempirei@256.bz
Portland, Oregon
San Francisco, California

May 7, 2012

Chapter 1

About

1.1 What is this about?

This is utterly pure and random bullshit directly from my mind. Other sources are referenced as they come[1] . The organization of this document is probably pretty poor. The reason this document exists is to make an attempt to record my pseudo-scientific thoughts in a central place and to dissuade myself from discarding everything and to attempt to transcribe and condense handwritten paper work before it is trashed or lost.

1.1.1 Current & Future Topics

Here is a list of current and future topics in no specific order. Further ordering may occur in a table of contents.

- Fast Cellular-Automata Wave Propagation
- Economics of Child Birth
- Analysis of 1-line Algorithmic Music in C
- Identifying Hierarchical Structure in Unordered Sets
- Password Cracking as a Cover Problem
- Observations of SMILES and Some Ideas
- Reverse-Compiling with Parametric Grammars

[1]Footnotes will primarily be used for referencing

1.1.2 About Me

I am Christopher Abad. I have worked extensively in Information Security for numerous companies since the late 90s. I have contributed to various minor projects, and have spoken at a number of insignificant hacker conferences. I was the owner of *20 GOTO 10*, an art gallery which I had the pleasure of operating from 2007 until 2009, in the Tenderloin neighborhood of San Francisco. An archival site for 20 GOTO 10 can be found at `http://www.twentygoto10.com/`. I can be contacted via email at `aempirei@256.bz` or `aempirei@gmail.com` and on AOL Instant Messenger via the screenname `ambientempire`. I also have a github account and all of my public repositories can be found at `https://github.com/aempirei`

1.1.3 Licensing

All of this random bullshit is being produced under the *DBARA*[2] license.

[2] "Don't be a retarded asshole" *(DBARA)* license.

Chapter 2

Fast Cellular-Automata Wave Propagation

Chapter 3

Economics of Child Birth

Evolution is generally efficient in that unnecessary processes seem to get pruned.

Because the pain and time components of birth have not been pruned, they must have purpose.

Because this purpose is unclear, understanding its purpose is prerequisite to understanding the birthing process and how it fits into the human model as a whole.

I make the assumption that pain and time of birthing can be modeled economically as a service.

Assuming that it is a service, an aspect of the service is the value of its service.

The measurement of the value of pain plays an important role in Tort for the recovery of losses due to pain and suffering through civil judicial procedure. I believe this sufficiently establishes that pain has real value and the expectation of remuneration after enduring pain is reasonable, and is likely essentially labor.

The time component should then be equivalent in principle to the time component in any economic service cost function with a time component, in that cost typically smoothly scales with time.

To approach understanding the purpose of the service, it may help to analyze the opportunity cost of the service.

True service may be unclear. Time and pain may be value or cost of hidden service. Opportunity cost of hidden service in a hypothetical market where goods and services and be purchased with time and pain is equivalent to set of goods and services that can be purchased with equivalent time and pain as close in nature as possible. If opportunity cost is narrower in breadth than opportunity cost afforded by equivalent value in plain money then sufficient correlation between alternatives to extract some understanding about the nature of the hidden service.

Here are various Categories I propose the study of cost. In all Categories the objects under study are Commodities, both goods and services. But the morphisms in each may create significantly different structures because the underlying value system or nomos are, as is hypothesized in Nomos & Narrative, irreducible with respect to each-other. Because of this, instead of a single Commodity Category, I've opted for a category per value system.

- Category of Opportunity Cost.
- Category of Money (Economic Capital).
- Category of Semiotics.
- Category of Purpose.
- Category of Cultural Capital.

Also consider the Nomos of the Bourdieu Field associated with the setting in which Pain as a captial exists. The aspect of this manner of characterization that is of specific interest is that field-specific nomos are considered irreducible with respect to another field and associted nomos, an Idea very reminicient of object isomorphism within a category. I believe the intent of Pierre Bourdieu's field concept and of Cover and Berger's Nomos was to intuitively begin to approach analysis of economics and sociology from a Categorical viewpoint. One criticism of Bourdieu's conceptualzation is the lack of conceptual clarity, leading to a wide variety of interpretations, mine being only the latest of many, but I believe, granted that I have interpreted the spirit of Bourdieu's work correctly, a Categorial approach would be a superior interpretation because clarity and specificity of concept is fundamental to Category theory. Additionally a rigorous treatment of Bourdieu's concepts can ease any previous concern for clarity, and although most mathematical frameworks do not mesh well with the social sciences, Category Theory has a natural indifference to the internal nature of the objects it studies. It is this indifference which is advantageous in the setting of social science where the objects are oft opaque to direct inspection.

Opportunity cost reflects the shortcomings of money isomorphism.

Real value of time and pain. Opportunity cost of real value equivalent for exposing the hidden value of a process otherwise unclear of purpose.

Chapter 4

Analysis of 1-line Algorithmic Music in C

This project was inspired by two youtube videos:

”Experimental one-line algorithmic music - the 2nd iteration” http://www.youtube.com/watch?v=qlrs2Vorw2Y

”Experimental music from very short C programs” http://www.youtube.com/watch?v=GtQdIYUtAHg

The musical generation of the 1-liner C program is of the form:

unsigned char transform(unsigned long t);

for(t = 0; t ¡ (1 ¡¡ N); t++) putchar(transform(t));

This also assumes tha output is used to feed a PCM audio device at 8000hz 8-bit mono.

This process can alternatively be described in many ways. It would first be useful to look at the loop iterator (t) in a specific novel way. What at first appears to be a simple monotonically increasing sequence may prove to be more useful when written as a set of binary vectors b_n each representing a single-bit slice of (t) over the entire sequence of iterations.

```
  t  =  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15 ... 2^N-1
     +-----------------------------------------------------------------------
b_0  |  0   1   0   1   0   1   0   1   0   1   0   1   0   1   0   1 ...   1
     +-----------------------------------------------------------------------
b_1  |  0   0   1   1   0   0   1   1   0   0   1   1   0   0   1   1 ...   1
     +-----------------------------------------------------------------------
b_2  |  0   0   0   0   1   1   1   1   0   0   0   0   1   1   1   1 ...   1
     +-----------------------------------------------------------------------
b_n  |  0 x (2^n) ...   1 x (2^n) ...                                       1
     +-----------------------------------------------------------------------
```

Taken independantly, each bit as an indepedant sequence is a square wave. Furthermore, each subsequent period is halving, and is incidentally orthagonal when taken as an entire set. These features readily form a very useful linear algebraic basis.

The signal frequency series is (don't forget nyquist):

```
f_n = frequency(b_n) (hz)
f_n = 8000 / 2^(n+1) (hz)
f_n = { 4000, 2000, 1000, 500, 250, 125, 62.5, 31.25, 15.625, ... } . (hz)
```

After about f_9, it is useful to consider the period of the vector.

```
p_n = 1 / f_n (s)
p_n = { ..., 0.128, 0.256, 0.512, 1.024, 2.048, 4.096, 8.192, 16.384, ... } . (s)
```

To select a tone and duration, it is simply a matter of selecting the corresponding bit vectors and composing the two in some manner. Selction is via an AND mask.

To select the 1000hz tone at bit vector b_2, calculate function b_2(t) = (t¿¿2)&1 To select a note duration of approximately 1/4th second we calculate:

```
       2^(n+1) = 8000 * 1/4
   n+1 log_2 2 = log_2 2000
           n+1 = log_2 2000
             n = log_2 2000 - 1
             n ~ 11 - 1 ~ 10
```

Then we select b_10 and calculate function b_10(t) = (t¿¿10)&1

An entire b_n vector could be composed by b_n = { b_n(0), ..., b_n(N-1) }

Now, the signals can be composed by simple binary amplitude modulation (AM) using either multiplication operation (*) or the bitwise AND (&) operation (provided that the bits are correctly aligned of course). We did happen to align them correctly above. This correct bitwise alignment could be done via normal matrix row-swap operations on the binary vector (t) instead of the bitwise shift operation (¿¿). Additionally, we should select a volume level. Assuming the output is unsigned 8-bit, we can simply shift the modulated signal by up to 7 positions to the left.

So we get the final signal generation function:

```
b_2_t = (t>>2) & 1   // select tone signal
b_10_t = (t>>10) & 1 // select duration signal
o = b_10_t & b_2_t   // modulate duration signal by tone signal
```

o ¡¡= 6 // select volume

Reduction of the above produces this compact form:

```
o = ( (t >> 2) & (t >> 10) & 1 ) << 6
```

Further reduction:

```
o = (t << 4) & (t >> 4) & 64
```

The same process can be described as a dot-product operation with (s), the selection vector, operating on (t) as s.t, and then another dot-product operation (c), the composition vector, also operating on (t) as c.t, and then finally the volume scalar (v), so that the complete modulation is v(s.t)(c.t).

At this point, this becomes an interesting point of study for Kolmogorov complexity and the computation of music.

Just given about 9 tones and 8 time intervals, it would be easy to sequence somewhat novel audio scores, but we're going to continue the breakdown of this basis to improve upon these numeric tools.

Given that the initial basis b_n is orthagonal, but of row rank significantly lower than its column rank, it would be of moderate interest to see if a complete linear independant orthagonal basis of rank 2N̂ could be generated from the original basis in some simple way. Given this, then in an albiet rather trival manner, any sequence of length 2N̂ could be managed in a rather long "1-liner". But then provided sufficient knowledge in the theory of data compression, an optimization between a minimal vector subset and a maximal signal approximation could be reached using only the most basic of algebraic operations.

Construction of a complete orthagonal basis of rank 2N̂ inituitively can be imagined to be by time-scaling of any of the original basis. One additional fact to remember is that each of the square-waves are self-similar and only differ in their time-scale. Also, provided that the total time interval of a complete sequence aligns on the period boundary of every basis vector, then the complete basis will retain orthagonality. So only a single representative vector from the basis is needed to generate an entire basis of rank 2N̂ provided a simple time-scaling function can be determined. Firstly, selection of the basis vector with minimal period is simply:

```
b_0(t) = (t & 1)
```

To scale a function in range by a constant factor of (k) would be k*f(n), while to scale a function in domain by the same factor would be f(n/k). So then it should follow that the period of the basis vector b_0 can be expanded simply by b_0(t/(k+1)) = (floor(t/(k+1)) & 1). The original b_0 square wave is an odd function, analogous to the sine function, so therefore given k = { 0, ..., 2N̂-1 } then the complete basis can be generated by:

```
b'_k = { b_0(2 * t / (k + 1)) } = { floor(2 * t / (k + 1)) & 1 }
```

With the frequency of each basis vector:

```
f'_k = frequency(b'_k) (hz)
f'_k = (8000/k) mod 8000 (hz)
f'_k = { 0, 4000, 2666, 2000, 1600, 1333, 1142, 888, 800, 727, 666, ... } . (hz)
```

Unfortunately, unlike the DST, the discrete implementations of this basis are not orthagonal because of aliasing problems at period boundaries of many of the square waves. When the wave frequency does not divide the number of discrete samples in the signal, then an imbalance in the distribution of the high and low states of the signal. Even tho this does not mean the basis is not linear independant, the basis loses some of the simplicities in decomposition calculations. Orthagonalization of this basis can be performed at the cost of significant complexity. A famous compromise between the hilbert square-wave basis and the advantages of orthagonality is the "Haar Wavelet".

Trying to revise the original modulation scheme with a target frequency of 2600hz, we get the following:

```
k = 8000/2600 ~ 3.08
b'_3.08 = b_0(2*t/(3.08+1)) = floor(2*t/(3.08+1))&1 = floor(t*2/4.08)&1 // select tone
b_10_t = (t>>10) & 1 // select duration signal
o = b_10_t & b'3.08  // modulate duration signal by tone signal
o <<= 6              // select volume
```

One may notice that when a frequency that is significantly misaligned with the total signal period is selected, many artifacts arise in the output signal.

Reduction of the above produces this compact form:

```
o = ( ((t>>10)&1) & (((int)floor(t*2.0/4.08))&1) ) << 6
```

Given a number of output signals o = {o_k}, these signals can then be composed together with a few possible methods, namely binary XOR $(\hat{})$, binary OR (—) and arithmetic addition (+), each with various auditory effects. Namely, addition runs the risk of clipping as the mean signal power approaches the range limit of the output channel (in this case 8-bit or 255), OR may saturate all of the output bits if the signal becomes too complex, and XOR may cancel out common-mode signals of similar power.

```
o' =  o_0 + o_1 + ... + O_n
o' =  o_0 ^ o_1 ^ ... ^ O_n
o' =  o_0 | o_1 | ... | O_n
```

More shit to come, obviously.

Chapter 5

Identifying Hierarchical Structure in Unordered Sets

Chapter 6

Password Cracking as a Cover Problem

Chapter 7

Observations of SMILES and Some Ideas

Chapter 8

Reverse-Compiling with Parametric Grammars

8.0.4 Enumerated Zots Candy

Various flavors of *Zots* can be enumerated.

1. Apple
2. Watermelon
3. Cherry
4. Grape
5. Orange
6. Lemon[1]

K^N or $\sum_{n=1}^{\infty} a_n$

[1]Item **??** ("Lemon") is disgusting