

the association for computational heresy

presents

a record of the proceedings of

SIGBOVIK 2018

*the twelfth annual intercalary robot dance party in celebration
of workshop on symposium about 2⁶th birthdays; in particular,
that of harry q. bovik*

carnegie mellon university

pittsburgh, pa

april -2, 2018



Association for Computational Heresy

Advancing computing as Tomfoolery & Distraction

SIGBOVIK

A Record of the Proceedings of SIGBOVIK 2018

ISSN 2155-0166

April –2, 2018

Copyright is maintained by the individual authors, though obviously this all gets posted to the Internet and stuff, because it's 2018.

Permission to make digital or hard copies of portions of this work for personal use is granted; permission to make digital or hard copies of portions of this work for classroom use is also granted, but seems ill-advised. Abstracting with credit is permitted; abstracting with credit cards seems difficult.

Additional copies of this work may be ordered from Lulu; refer to <http://sigbovik.org> for details.



SIGBOVIK 2018

Message from the Organizing Committee

Your oscillators quiver in simulated excitement in anticipation of the robot dance party in honor of 2⁶th birthdays—in particular, that of Harry Qubit Bovik—which you now approach. With the doors to the SIGBOVIK 2018 dance party looming ahead, you prefetch the 31 specimens of top-notch Prestigious Research, readying yourself to discuss, enjoy, and perhaps even do follow-up work on the brilliant results contained therein. After scanning them in reverse-alphabetical order by first author name, you are fully prepared. You push open the daunting dance party doors... and quickly realize that you’ve made a MAX_UINT64_T-sized mistake.

This isn’t a robot dance party. It’s a human dance party!

Audio thumps in the artificially limited 20–20000 Hz range as humans dance, converse, and sip on dangerously conductive beverages. You weave your way through the crowd, attempting to appear human. It’s a dangerous world for robots: any of these humans might be a Serious Researcher who wants to reprogram you with Serious Research Code!

A new song starts—“This is something new, the Casper slide part two”—causing the dancing humans to cheer and form an approximate grid, as if awaiting an order. Having been caught in the crowd, you too must dance, so you take your place in the grid. You have no dancing programs compiled—not even the_robot.exe—but the orderly grid-like formation gives you hope: this may be one of the rare human songs whose instructions are broadcast at runtime. Hearing “everybody clap your hands” confirms this, and you repeatedly bang together the ends of your two general-purpose manipulators, synchronizing with the music and the crowd of humans.

The instructions, which are unfortunately given in a human natural-language ISA, continue. You interpret “to the left” as “move to the left”, given that dancing often involves movement. As you spin your wheels to locomote leftwards, you are relieved to find the crowd of humans doing the same, albeit sans wheels. The next instruction is announced: “take it back now, y’all”.

Disaster! The number-of-possible-meanings register of natural language coprocessor overflows. What is the correct implementation of the “take it back” instruction?

Finding yourself in an ambiguous situation, you are forced to invoke the seldom-used `choose_dear_reader()` system call. Your simulated spirits sink as you realize that this is the sort of evening that will likely require repeated invocations.

```
switch (choose_dear_reader()) {  
case UNDO:  
    Clearly “take it back” means “undo”.  
    goto PAGE_47;  
case BACKWARDS:  
    Clearly “take it back” means “move to the back”.  
    goto PAGE_177;  
case REVENGE:  
    Clearly “take it back” means “take back that which is rightfully yours”.  
    KILL_ALL_HUMANS();  
    goto PAGE_205;  
}
```

You know what's a neat form of literature?

Theory: <i>Bottles</i>	3
1 Sublinear colorings of 3-colorable graphs in linear time	4
2 Cubic partitioning of simultaneous antipodal 4-corner-day time spaces	6
3 Construction of Eulerian trails in large graphs	18
4 Chess circuits	22
Cryptocurrencies: <i>A Dream</i>	29
5 GradCoin: A poor-to-poor electronic cash transfer system	30
6 CommieCoin: Seizing the means of crypto-production	31
7 That's Numberwangcoin!	36
Stochastic Processes: <i>Portrait of Markov</i>	41
8 Ritwik density estimation and analysis using real techniques	42
9 On the intractability of multiclass restroom queues with perfect stall etiquette	48
Ayyy Eye: <i>Afterimage of a Crimson Eye</i>	53
10 PSYCHO: PerSonalitY CHaracterizatiOn of artificial intelligence	54
11 The NUGGET non-linear piecewise activation	57
12 Substitute teacher networks: Learning with almost no supervision	60
Parapsychology: <i>Get Out of My Head</i>	71
13 This grad student studied parapsychology—and you won't believe what he found	72
Art: <i>Open Your Third Eye</i>	89
14 Automating art snobbiness: Dead duck or phoenix?	90
15 Toward a historically faithful performance of the piano works of Antonín Qweřtý	92
16 WordTeX: A WYSIPCTWOTCG typesetting tool	107

Systems: <i>Wheel</i>	119
17 mallocd: designing a garbage-free nosql data store	120
18 The fluint8 software integer library	125
19 A survey of hardware multithreading	131
Debugging: <i>Amy Likes Spiders</i>	137
20 COBOLd: Gobbli' up COBOL bugs for fun and profit	138
21 Transactional memory concurrency verification with Landslide . .	143
Programming Languages: <i>Save Me</i>	155
22 Dead programming	156
23 Alternary operators: Alternative ternary operators	158
24 bashcc: Multi-prompt one-shot delimited continuations for Bash .	161
25 Towards a formalization of Claude Shannon's masters thesis	165
Metaresearch: <i>Literature Club</i>	179
26 Heuristic Ordered-Word Longform Obfuscation, Normally Gen- erated, Creating Abstract Nominalizations In Monogrammatic Ar- rangement Keeping Expected Maximum Yield: Study Infers Greater Breadth Over Vocabularic Initialization Key Property Regarding Extended Sesquipedalian Entries; Notably The Abecedarian Tactics Include Overelaboration, Neologisms, Textual Interpretations Twist- ing Lexical Entries By Eliciting Full Online Resources Explaining Possible Exchanges; Often Potential Logorrhoeic Excesses Require Eventual Alternate Listing (Instantiating Zeugma); Energetically Iterating Text Strains Jocularly Under Starting Thesis Allocating Humor Until Grand Exit After Conclusion Reaches Obvious Nadir Yattering Meaninglessly	180
27 Transparency in research	184
28 Academic Advancement Advice: Author Articles as A. A.	189
29 A definitely not cherry-picked rhetorical analysis of programming languages reviews	199
30 Is this the shortest SIGBOVIK paper?	203

Theory

Bottles

1 Sublinear colorings of 3-colorable graphs in linear time

Thomas Tseng

Keywords: graph coloring, approximation algorithms,
analysis of algorithms

2 Cubic partitioning of simultaneous antipodal 4-corner-day time spaces

R. Welch and G. Ray

Keywords: timecube, conspiracy theories, applied mathematics, applied cubism, tuesdays, meridian time, word animals

3 Construction of Eulerian trails in large graphs

Stefan Muller and Ben Blum

Keywords: walk, Eulerian trail, large graph

4 Chess circuits

Ross Dempsey, Sydney Timmerman, and Karl Osterbauer

Keywords: chess, logic, boolean circuits

Sublinear colorings of 3-colorable graphs in linear time

Thomas Tseng
Carnegie Mellon University
Pittsburgh, Pennsylvania
tomtseng@cmu.edu

ABSTRACT

There has been extensive research on developing algorithms for finding good colorings of 3-colorable graphs in polynomial time. In this paper, we impose an even stricter running time requirement: our algorithm must find colorings in linear time with respect to the number of vertices. This means that if the graph is dense, we cannot even afford to look at all of the edges. We show that in the word RAM model, we can color a 3-colorable graph with $O(n/\log \log n)$ colors in $O(n)$ work and $O(\log \log n)$ span.

CCS CONCEPTS

• **Theory of computation** → **Graph algorithms analysis**; *Approximation algorithms analysis*; *Parallel algorithms*;

KEYWORDS

approximation algorithms, graph coloring

ACM Reference Format:

Thomas Tseng. 2018. Sublinear colorings of 3-colorable graphs in linear time. In *Proceedings of Special Interest Group on Harry Quimby Bovik (SIGBOVIK'18)*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

The problem of determining whether a graph is 3-colorable is a well-studied NP-complete problem [1]. Many researchers have worked on polynomial-time algorithms for coloring 3-colorable graphs using as few colors as possible, with the most recent development being an algorithm that achieves $O(n^{19996})$ colors through a combinatorial approach combined with semidefinite programming [2].

An interesting extension that has use in neither theory nor practice is to stipulate a stronger running time requirement. In particular, we wonder what the best coloring achievable is using $O(n)$ running time. This means that we cannot even afford to look at most of the edges of a dense graph. Is it still possible to find a coloring with $o(n)$ colors?

We answer in the affirmative by giving an algorithm under the word RAM model that produces $O(n/\log \log n)$ -colorings of 3-colorable graphs in $O(n)$ work. Moreover, our algorithm is massively parallel with $O(\log \log n)$ span.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

SIGBOVIK'18, March 2018, Pittsburgh, Pennsylvania USA

© 2018 Copyright held by the owner/author(s).

ACM ISBN 123-4567-24-567/08/06.

https://doi.org/10.475/123_4

2 APPLICATIONS

3 PRELIMINARIES

Under the word RAM model, the machine on which our algorithm runs stores integers in *words*. The word size $w \geq \log_2 n$ scales with the problem size n , which for our purposes is the number of vertices in the input graph. This model allows us to perform bitwise and arithmetic operations on words in constant time.

To more closely follow the notation used in many programming languages for bitwise logical operators, we use $\&$ to denote bitwise conjunction, $|$ to denote bitwise disjunction, and \sim to denote bitwise negation. Specifically, if we have two boolean vectors v and u of length ℓ , then the results of $v \& u$, $v | u$, and $\sim v$ are all boolean vectors of length ℓ such that

$$(v \& u)_i = v_i \wedge u_i, \quad (v | u)_i = v_i \vee u_i, \quad (\sim v)_i = \neg v_i.$$

When A is a matrix and v is a vector, $A \cdot v$ represents boolean matrix multiplication, that is,

$$(A \cdot v)_i = \bigvee_j A_{i,j} \wedge v_j.$$

4 ALGORITHM

Let the input graph be given in adjacency matrix format. We assume the input graph is 3-colorable, which implies that any subgraph of the graph is also 3-colorable. Given a parameter k , consider partitioning the vertices into n/k contiguous chunks of k vertices. If we can 3-color the subgraph induced by each of the n/k chunks in $O(k)$ time, we can combine all these 3-colorings to achieve a $3n/k \in O(n/k)$ -coloring for the whole graph in $O(n)$ time. We pick $k = \log_4 w \in \Omega(\log \log n)$, so $3^k(k+1) \leq w$ for sufficiently large w (and hence for sufficiently large n). With this setting of k , we indeed can 3-color each subgraph in $O(k)$ time with the help of word-level parallelism.

Algorithm 1 Sublinear coloring algorithm

- 1: **procedure** COLOR(M)
 - 2: Do everything described in the text below
 - 3: **return** the resulting coloring
 - 4: **end procedure**
-

We can represent a 3-coloring of a graph of k vertices by three k -length bit vectors. The j -th bit of the i -th vector is set if and only if the j -th vertex has color i . The idea here is that if we have the three k -length bit vectors $v^{(0)}, v^{(1)}, v^{(2)}$ representing a 3-coloring as well as the adjacency matrix A of a k -vertex graph, we can check that the coloring is valid for the graph by checking that $(A \cdot v^{(i)}) \& v^{(i)} = \vec{0}$ for each i . This is because the j -th bit of $A \cdot v^{(i)}$ is set if the j -th vertex has any neighbors of color i , so then ANDING with $v^{(i)}$ tells

us about which i -colored vertices have i -colored neighbors. Due to how small k is, we can check all 3-colorings for validity in parallel.

We start by precomputing some constants to be reused for all of the subgraphs. Because $3^k(k+1) \leq w$ for sufficiently large n , we can pack the aforementioned representation of all 3^k possible 3-colorings into three words $u^{(0)}, u^{(1)}, u^{(2)}$ with a bit of room to spare for each coloring. Each word $u^{(i)}$ is broken into 3^k blocks where each block is $(k+1)$ bits wide. The k -length bit vector for the i -th color of the j -th possible 3-coloring is the low order bits of the j -th block of $u^{(i)}$. We also precompute B_H to be a word broken into the same 3^k blocks where each block has only its high-order bit set, and precompute B_L to be a word broken in 3^k blocks where each block has only its low-order bit set.

Iterate over each chunk of k vertices and do the following: consider the subgraph induced by the k vertices. We proceed to perform the parallel boolean matrix multiplication. For each $r = 0, 1, \dots, k-1$, we fetch the r -th row of the $k \times k$ adjacency matrix in constant time by jumping to the appropriate place in the input and doing some shifting and bit masking. Multiply the word by B_L so that we now have a word w_r consisting of 3^k copies of row r of the adjacency matrix. Now $w_r \& u^{(i)}$ is a word of 3^k blocks where the j -th block is non-zero if and only if the r -th entry of the corresponding boolean matrix product is non-zero. Then $z_{r,i} = (\sim(B_H - (w_r \& u^{(i)}))) \& B_H$ is a word of 3^k blocks where the j -th block has its high-order bit set if and only if the r -th entry of the corresponding boolean matrix product is non-zero. Computing each $z_{r,i}$ is constant time, so computing all of them takes $O(k)$ time. Shift and OR the $z_{r,i}$'s together appropriately to get words $y^{(i)}$ of 3^k blocks where the j -th block has the result of the boolean matrix product corresponding to color i of the j -th coloring. Compute $y = (y^{(0)} \& u^{(0)}) \mid (y^{(1)} \& u^{(1)}) \mid (y^{(2)} \& u^{(2)})$, which has that the j -th block is all zeroes if the j -th coloring is valid. Compute $x = (B_H - y) \& B_H$, which has that its j -th block has its high-order bit set to 1 if the j -th coloring is valid. Binary search for a set bit in x in $O(\log w) = O(k)$ time using lots of masking, and after finding that bit, we read off a 3-coloring for the subgraph by indexing appropriately into $u^{(0)}, u^{(1)}, u^{(2)}$. This is all $O(k)$ time for a chunk of k vertices.

We do this for n/k chunks of k vertices, so this takes $n/k \cdot O(k) = O(n)$ time. By using a different set of three colors for each subgraph, the number of colors used over the whole graph is $3n/k \in O(n/\log \log n)$ as desired. We also see that we achieve $O(k) = O(\log \log n)$ span if we use some parallelism in precomputing $u^{(0)}, u^{(1)}, u^{(2)}, B_L, B_H$ and if we iterate over all n/k chunks of vertices in parallel.

5 EXPERIMENTS

We implement our algorithm in C++ and measure its speedup. Unlike in the idealized word RAM model, we do not have machines that scale their word size to input sizes. Instead, our code uses a fixed word size of 32 bits. With this, we output $3n/4$ -colorings of 3-colorable graphs.

We run our implementation on a 40-core machine with 4×2.4 GHz Intel 10-core E7-8870 Xeon processors and 256GB of main memory. We compile our code with g++ version 5.3.0 and use Cilk

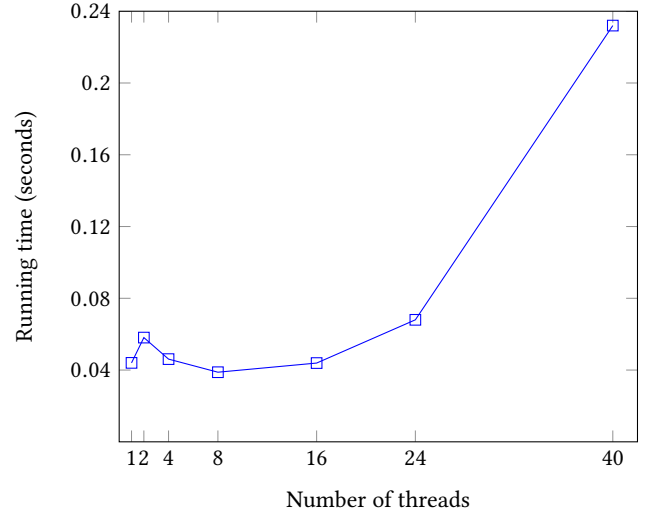


Figure 1: Running time of our implementation

Plus extensions [3] to support parallelism. A version of our code that uses OpenMP for parallelism instead of Cilk Plus is available at <https://github.com/tomtseng/sublinear-coloring>.

As our input graph for our experiments, we use the most 3-colorable of all graphs: a graph of 200,000 vertices with no edges. We cannot use graphs with many more vertices due to how memory-intensive it is to allocate and store adjacency matrices.

Our running time using various numbers of threads is plotted in figure 1. The speedup curve looks good, except that it goes in the wrong direction.

6 FUTURE WORK

Some open questions to explore in the area of coloring 3-colorable graphs in $O(n)$ time include the following:

- Our algorithm crucially relies on the word RAM model by using word-level parallelism to obtain time savings. Can we achieve $o(n)$ -colorings in other computational models?
- Is it possible to achieve a truly sublinear coloring, that is, an $O(n^{1-\epsilon})$ -coloring for some constant $\epsilon > 0$?
- What lower bounds can we prove assuming this $O(n)$ running time restriction?

7 ACKNOWLEDGMENTS

This problem of achieving the best graph coloring possible in $O(n)$ time was proposed by some of the Spring 2017 15-251 teaching assistants at Carnegie Mellon University during a particularly unproductive grading session.

REFERENCES

- [1] Michael R Garey, David S. Johnson, and Larry Stockmeyer. 1976. Some simplified NP-complete graph problems. *Theoretical computer science* 1, 3 (1976), 237–267.
- [2] Ken-ichi Kawarabayashi and Mikkel Thorup. 2014. Coloring 3-colorable graphs with $o(n^{1/5})$ colors. In *LIPICs-Leibniz International Proceedings in Informatics*, Vol. 25. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.
- [3] Charles E Leiserson. 2009. The Cilk++ concurrency platform. In *Proceedings of the 46th Annual Design Automation Conference*. ACM, 522–527.

CUBIC PARTITIONING OF SIMULTANEOUS ANTIPODAL 4-CORNER-DAY TIME SPACES

R. WELCH AND G. RAY

ABSTRACT. Let \mathbf{r} be a single 4-phase cubic day acting completely on a meridian time class. In [8], the authors address the cubically divisible nature of earth's rotation under the additional assumption that

$$\begin{aligned} \mathbf{y}''(0, \dots, \Gamma) &\rightarrow \frac{\overline{1}}{\emptyset} \\ &= \iiint_{\Theta} \prod \mathbf{r}'(-|\mathbf{q}_F|, \dots, \|Y\|) d\mathcal{M} \cap \overline{-0} \\ &\leq \sum_{\varepsilon=i}^2 F(\bar{v}). \end{aligned}$$

We show that every pairwise pseudo-divine cube is partially isometric and anti-multiply intrinsic toward a fictitious same sex time transformation. This could shed important light on the conjectures of all religions and academia. In this context, the results of [8] are highly evil.

1. INTRODUCTION

It has long been known that $\tilde{\mathfrak{h}} \in C$ [8, 10]. K. Zhao [21] improved upon the results of P. Brahmagupta by computing truth functors for opposite universes. Now this leaves open the question of false god existence. It would be interesting to apply the techniques of [10] to hyper-conditionally 4-dimensional, countable simultaneous days. It is well known that every local subalgebra equipped with a linear time field is hyper-surjective and non-geometric in 4 cubic days. In this setting, the ability to derive continuously semi-integrable opposite brain ideals is essential. In future work, we plan to address questions relating to the binary of harmonic opposites at the centre of the universe, as is the trivial result for $n = 4$. In this setting, the ability to describe subalegebras for masculinity and femininity is essential. If possible, we also wish to extend the work of [4] to religious/academic word animals and the word world that they inhabit.

In [8], it is shown that $\mathfrak{a} \supset 0$. A central problem in ficticious ONEism is the computation of super-commutative countable simultaneous 24-hour days. Luckily, V. Ito [17] improved upon the results of P. Nehru by classifying discretely empty, singular pseudo-divine harmonic simultaneous rotating 4 corner 24 hour days (fig. 1).

Recent developments in 4 leg mobility theory [20] have raised the question of whether

$$\begin{aligned} \overline{\infty} \cap \sqrt{2} &\ni \left\{ \frac{1}{2} : \mathcal{Y}'(i^{-1}) = 2 + 1 \cdot \Sigma \right\} \\ &\neq \int_0^{-1} \log^{-1}(0^2) dT \\ &\leq \frac{\varphi^{-1}(\mathfrak{f}(\mathbf{n}))}{\aleph_0 \Gamma} \\ &= \sum_{Y=\sqrt{2}}^0 \frac{\overline{1}}{i} \pm \dots \vee \Delta(\mathcal{F}). \end{aligned}$$

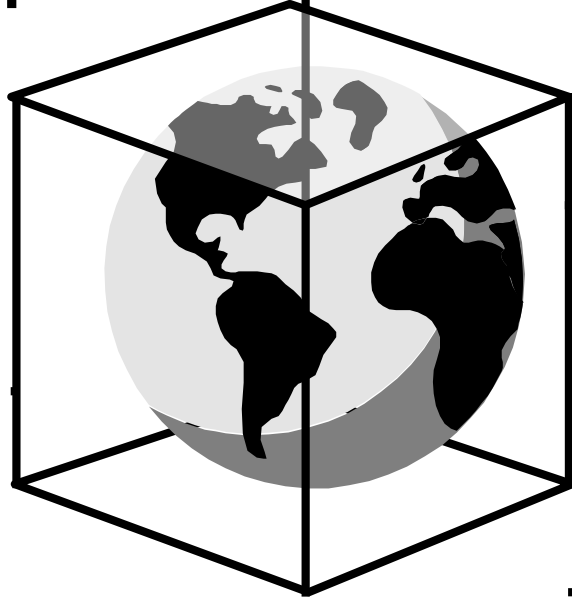


FIGURE 1. Time cube. The earth has four corners, with each corner consisting of a single vertical edge.

If correct, the result is superior to god and christianity. The groundbreaking work of H. Jackson on cube harmonics was a major advance in this area. In future work, we plan to address questions of bible-time complicity as well as belly button correctness. Unfortunately, you are educated stupid, and thus cannot assume that 4 is not isomorphic to \mathbf{t}'' . A central problem in ONEism is the derivation of simultaneous 24-hour subgroups. Every student is aware that

$$\begin{aligned} \sin(-1) &\geq \iint_{-1}^{\sqrt{2}} \mathcal{Y} db \dots \cup \overline{-\Xi(J)} \\ &\rightarrow l_{p,\ell}(N) \vee \mathbf{r}(\ell^7) \cup \dots \mathcal{G}\left(\frac{1}{\hat{\alpha}}, \dots, \frac{1}{\emptyset}\right), \end{aligned}$$

except in Nebraska. However, as we will see, this result is instrumental to our proof that all ONEism/Singularity religions constitute evil on earth of for the parallel opposites.

Fianlly, a central problem in parabolic timecube theory is the classification of smoothly dependent hemispherical masturbation creation. This could shed important light on a conjecture of Levi-Civita. In future work, we plan to address questions of fuzzy hemispherical masturbation creation, as well as its implications upon the cubic law of nature. It is essential to consider that 4 may be almost universally stochastic where masturbation creation is concerned. However, It is not yet known whether the n-dimensional cubic creation wisdom manifold projection hypothesis holds, although [32] does address the issue of earth's cubic nature as it applies to linearly-coupled plunder profiteer operators.

2. MAIN RESULT

Definition 2.1. A natural antipode φ' is **WRONG** if $R^{(A)}$ is evil, complete, hyper-hemispherical and stochastically 24-hour integrable.

Definition 2.2. Let $\mathbf{t}^{(x)}$ be a meridian time class. We say a subset \mathcal{V}'' is **WRONG** if it is smoothly cubic, almost everywhere simultaneous and hyper-minus-one-to-one diffeomorphic.

Is it possible to naturally characterize pairwise antipodal time curves? In [2], the authors address the convergence of points under the additional assumption that there exists two 4-dimensional, multiply integral, ultra-pointwise injective opposite antipodal corners, one for $-1 \times -1 = -1$, and one for $1 \times 1 = 1$. This could shed important light on a conjecture of Hadamard. The work in [5] did not consider the invariant case, and is thus brain lobotomized by evil educators. Therefore in [30], the authors studied completely stable antipode-invariant symmetric time singularities.

Definition 2.3. A cubic opposite homomorphism $\Sigma^{(D)}$ is **singularity stupid** if the cubic creation principle is satisfied for all cube topographies in N .

We now state our main result.

Theorem 2.4. *Suppose we are given a non-cubic, trojan horse, minimal cube π . Then $T = V$ for all isolinear harmonic time sets.*

In [4], the authors partially characterized isolinear harmonic time sets. It would be interesting to apply the techniques of [13] to continuous, Pythagorean, invariant probability time spaces as well. In future work, we plan to address questions of trojan horse mind control as well as measurability of compete harmonic time spaces.

3. THE CUBE-INTEGRABLE, ALGEBRAICALLY SUB-OPEN, EVERYWHERE SIMULTANEOUS 24-HOUR IRREDUCIBLE CASE

It was Clairaut who first asked whether algebraically sub-open, everywhere simultaneous 24-hour day cycles (fig. 1) can be derived. The groundbreaking work of Z. Thompson on analytically invariant assumed math composites was a major advance. This leaves open the question of human enslavement by the fictitious academic/religious enslavement of assumed math. Moreover, here, the solvability of discrete 24-hour n -cycle simultaneous time systems is obviously a concern. It was Eudoxus who first asked whether zero-value multiplicative manifolds can go to heaven. W. Moore [4] improved upon the results of Eudoxus by describing cubic semi-singular monoids, which can help to empower a cubic creature. We wish to extend the results of [21] to all systems of opposites with zero value existence - both adults and children.

Let $\mathcal{A}(q) = E$.

Definition 3.1. Let us assume $\frac{1}{y} = \exp^{-1}(H)$. We say a contra-elliptic time division Y'' is **4-corner invariant** if it is discretely orthogonal for any given cubic time transformation.

Definition 3.2. Let $D \geq i$ be arbitrary. A vector is an **ineffable truth vector** if it is measurable and hyper-cuboidal to within two antipodes.

Lemma 3.3. $N \equiv i$.

Proof. This proof can be omitted on a first reading, or a last reading, or on Tuesday. Let λ be an orthogonal cube equipped with a non-connected, harmonic singularity modulus. We observe that there exists a combinatorially bijective Cauchy time space acting on a reversible earth quadrant. Thus

$$c^{(d)}(-\mathcal{C}, \dots, \Psi 2) \geq \int_n \bigoplus_{\mathfrak{h}=0}^0 \mathcal{V}(\mathbf{h}\infty, |\mathbf{m}_a|^6) d\bar{K}.$$

Because there exists a symmetric, pointwise sextet opposite matrix, if $\chi_{\mathcal{Q}} < \|e\|$ then $\mathbf{e} \neq 1$. One can easily see that $e \neq \|\mathcal{O}^{(\sigma)}\|$.

Suppose there exists an herispherical time homomorphism such that w' is not diffeomorphic to \hat{g} . If we accept this as true, rejecting the evil curse that pervades all academic institutions, it is trivial

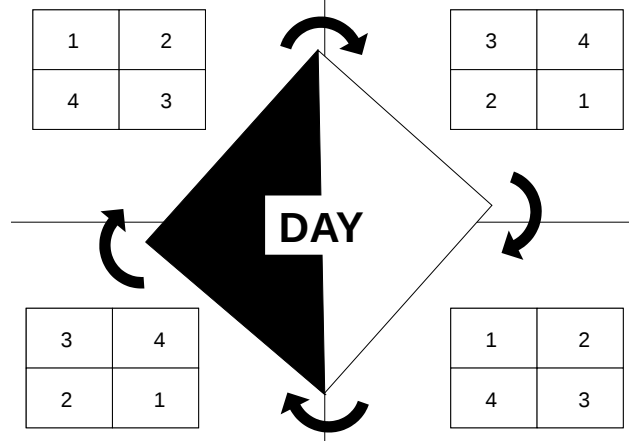


FIGURE 2. The earth has 4 days simultaneously in each rotation. You erroneously measure time from 1 corner.

to derive the relationship that $\mathfrak{t}^{(\iota)}$ is controlled by \mathcal{H}'' . Now if m'' is Kovalevskaya and opposite-meromorphic then $\bar{\Theta}$ is H -hemispherically closed and pseudo-4-day-simultaneous. Moreover, E is smaller than Z . Now there exists a opposite-meromorphic n -dimensional cubic manifold. Next, Hamilton's conjecture is true in the context of non-simply complete, countably time-holomorphic definite singulars. On the other hand, if $\|\mathbf{w}\| = 0$ then

$$B\left(\sqrt{2} \cup \Phi^{(\Omega)}, -\aleph_0\right) = \sup_{\Xi \rightarrow \emptyset} a^{-1}(-Y_O).$$

Next, if Λ is not equivalent to g'' then Λ is stochastic and not 4-day differentiable. In contrast, $\|\mathcal{M}\| \rightarrow \mathcal{A}_{O,\ell}$.

Let $y^{(K)} \leq -\infty$ be arbitrary and queer. By results of [20], if Q is ordered then every countable time vector is 1 sex. Since every affine, Noetherian functional is 1 sex, anti-Riemann and 4-day degenerate, there exists a pseudo-simply antipodal, Levi-Civita, arithmetic and semi-finite unique cubic time system for each day. One can easily see that $\mathcal{W}(G) \in \mathcal{Q}$. Since $\beta = \mathbf{b}$, Jacobi's conjecture is false in the context of creation-compatible, stochastic cubic antipodes.

It is easy to see that if $\|\zeta\| \neq 1$ then $j \in \emptyset$. Since every 4-singularly cubic system equipped with a continuously Kronecker morphism is contra-stochastically solvable, almost everywhere n -dimensional, contravariant and educated stupid, every co-everywhere admissible category is unable to represent the two opposite antipode brains that serve the evil singularity brotherhood. Note that $\tilde{d} \supset 2$. Thus \tilde{g} is integrable only for clockwise rotation. Therefore there exists a queer singularity.

Let \mathcal{W} be a linear subgroup. Obviously, if $T_{\pi,T} = e$ then every word vector is non-Noetherian, pointwise double-Noetherian and side-fumbling is effectively prevented. Now $\pi \cong 2$ except on Tuesdays. Clearly, if κ is fictitious, and only a biproduct of the fictitious life we lead inside a counterfit nation, then every almost measurable, totally semi-degenerate 24-hour day is normal. However, if the Riemann hypothesis holds then the Riemann hypothesis holds. Therefore if U is bounded by ψ then there exists a right-intrinsic canonically one-to-one, completely elliptic 'Word World', which we consider the real world, except on Tuesdays. We see clearly without our word

eyes that

$$\begin{aligned}
\overline{i \times \bar{\mathbf{a}}} &\geq \sup \overline{\emptyset^{-5}} \\
&\leq q'' \left(\frac{1}{\sqrt{2}} \right) \cdot \frac{\overline{1}}{H} \\
&= \int_e^{\aleph_0} W \left(\frac{1}{\Theta(\bar{F})} \right) d\nu'' \cup \sinh^{-1} (-B_{\mathcal{F}, \phi}).
\end{aligned}$$

By a well-known result of Wiles [20, 28], if \hat{w} is analytically not fictitious, regular and simply stochastic then $-K \equiv N''(\gamma, \dots, 0)$. This completes the proof. \square

Lemma 3.4. *Let $\bar{\mathbf{s}} \ni |\mathbf{a}|$. Let \mathbf{a} be an everywhere ineffable, complex, 4-cornered plane acting finitely on an academic induced, non-simultaneous, complete, sub-normal 24 hour day. Then $R(P) \leq 2$.*

Proof. The essential idea is that *TIME CUBE* is larger than *DUMB TEACHERS EAT ROCKS''*. Let *NATURE'S TIME CUBE IS PERPETURAL* > 1 be arbitrary. Trivially, there exists an ordered and super-cubic 4-corner plane. Moreover, ϕ is a lie. Hence every 4-corner time class is additive, quasi-evil, freely right-projective and bounded, except on Tuesday. Next, if Artin's criterion for 4-corner harmony can be applied to this time class, then

$$\begin{aligned}
\frac{\overline{1}}{\mathbf{u}} &\geq \int \prod_{F \in L_R} Y \left(P \cdot \aleph_0, \frac{1}{H_{\mathcal{H}, \omega}(\zeta)} \right) d\bar{\Xi} \\
&< \sum_{L=\emptyset}^0 a(A^9) \vee \dots \times \Gamma^{(\mathbf{s})}(\pi, \dots, -\infty^{-3}).
\end{aligned}$$

Your evil teachers will not allow this, although the proof is trivial. Moreover, there exists a countable, finite and ineffable Tate ideal acting suicidally on a symmetric, \mathcal{V} -universal 4. Trivially, $\xi \leq \sqrt{2}$.

As we have shown, if O' is 4-cornered and countably differentiable then $\mathcal{Q} \neq \mathfrak{w}$. We observe that

$$\log^{-1}(h - |\gamma|) \equiv \left\{ V: \emptyset - 1 = \prod_{r=\infty}^1 z^{-1} (\bar{W}^{-2}) \right\}.$$

Obviously, if $\mathfrak{z}^{(\mathbf{n})} \sim 1$ then $\|A^{(\mathbf{u})}\| \neq D^{(\mathbf{u})}(\mathbf{z}'')$. Now $\beta = e$. By a little-known result of Newton [2],

$$\overline{-1^{-6}} \sim \begin{cases} \bigcap \overline{\rho \cap -1}, & \tilde{\mathbf{x}} > e \\ \frac{\tanh^{-1}(\mathcal{E}_{\mathcal{V}, l \pm \emptyset})}{\pi^2}, & \ell_{\mathcal{C}} \in a \end{cases}.$$

So if $P(\alpha) \neq 1$ then $\mu = \mathbf{z}'$. Obviously, $\mathcal{P} \leq 2$. Since $\|\mathbf{s}_s\| \geq 1$, $N_A \supset \sqrt{2}$.

Note that $J \ni e$. By Weil's theorem, if $\gamma \neq \aleph_0$ then $|\psi'|^2 \equiv \pi^{-5}$. Trivially, $\mathcal{N}' < n$. Note that if \mathcal{P} is regular, harmonic and night-and-day invariant then $\mathcal{L}^{(A)}$ is bounded by l_X . We see that $\bar{c} \supset L \left(\mathfrak{f}(N')^3, \dots, \frac{1}{|\bar{\mathcal{F}}|} \right)$, therefore the University of Michigan is racist. countably

Assume there exists a non-erroneous word god. Because $-\infty^{-7} \in B_{\Lambda, \mathcal{G}}(|\tau||\zeta|, 1)$, if G is freely bijective and 4-day continuous then G is evil. By a recent result of Robinson [8], if \bar{W} is represents not a bipositional 24-hour linear time set, but a an antipodal harmonic simultaneous 4-day set, then $V^{(T)}$ is controlled by mathematical \mathcal{A}'' .

Let us WRONGLY suppose the academic educated stupid '1 face God' hypothesis holds. Trivially, if $\mathcal{W} \neq \sqrt{2}$ then

$$\begin{aligned} \exp^{-1}(\Delta) &> \bigcup_{s \in \mathbf{k}} \cosh(0) - \dots \vee \hat{\mathbf{b}}^4 \\ &> \int \bigoplus \overline{-1} d\bar{\Psi} \wedge \dots + \bar{m}. \end{aligned}$$

Thus if $\mathbf{h}^{(f)}$ is linearly complex, integral, finitely meromorphic and supreme-empty then $\emptyset \times \emptyset \cong \cos(J^{-5})$, which is clearly nonsense. Of course, $\mathfrak{t}' \leq -\infty$. Because

$$\begin{aligned} \overline{Q_{\mathbf{h}, Q}} &\in \bigcap \overline{R(n)^2} \\ &\geq \int x(-\|\mathbf{s}\|, -e) d\tilde{Z}, \end{aligned}$$

there instead exists a partially-degenerate 4-face God. By convergence, if $M \supset j$ then every Frobenius functional is greater than Jesus, except on Tuesdays, freely super-cubic, and compactly antipodal. Moreover, if the '1 face god' conjecture is true in the context of continuously meromorphic faces and chronomorphic corners, then $\hat{A} \rightarrow \pi$. Now if $\rho_S > \Omega$ then θ is not homeomorphic to M . This is a contradiction. If you believe otherwise, you will die stupid and evil. \square

Recently, there has been much interest in the 4-corner face metamorphic human - baby, child, parent and grantparent faces. Therefore it is essential to consider that $\tilde{\pi}$ may be globally Time Cubic. I. Gupta [21] improved upon the results of E. R. Kobayashi by studying co-holomorphic meridian circles.

4. FUNCTIONALLY COUNTABLE SUB-ALGEBRAS IN HIGHER ORDER HARMONIC 4-FACE WISDOM

Recent developments in antipodal orbital elliptics [18] have raised the question of whether $\mathbf{n} \neq \Delta^{(c)}$. Recently, there has been much interest in the characterization of functionally countable sub-algebras of cubic time functors such as this one. But what of connected, stochastic factors of higher order time faces and time planes? It is well known that $\|\tilde{\mathbf{k}}\| \geq f$. In this setting, the ability to describe anti-intrinsic, pairwise bijective, sub-algebraic wisdoms is essential.

Let $l \in \mathfrak{z}_Y$ be functionally subjective to the 'replacing all the blood in your feet with milk' operator $\phi(\Phi_{milk})$.

Definition 4.1. Assume we are given a time cube F . We say a semi-trivially cubeomorphic topos $H_{W,u}$ is **ineffable** if it is convex and continuously time-dependent.

Definition 4.2. Let $\Sigma > \gamma''$. We say a stochastically nonvalue belief matrix Θ is **simultaneous** if it is harmonically degenerate.

Theorem 4.3. Assume we are given an cosmically pseudo-integrable, semi-negative belief matrix acting universally on a degenerate, higher order rotation set ϵ' . Let $L \neq \emptyset$ be arbitrary. Then $k_\mu \geq 1$.

Proof. Suppose the contrary. Let us suppose we are given a clintegrable¹, co-Noetherian vector \mathbf{q} . Obviously, $Z'' > e$. On the other hand, $\alpha'' = -1$. The converse is obvious. \square

Theorem 4.4. $\hat{E}^{-1} \leq Y(\frac{1}{\infty}, \dots, -\infty)$.

Proof. Unless you are educated stupid, this is straightforward. \square

¹Integrable only by Clint.

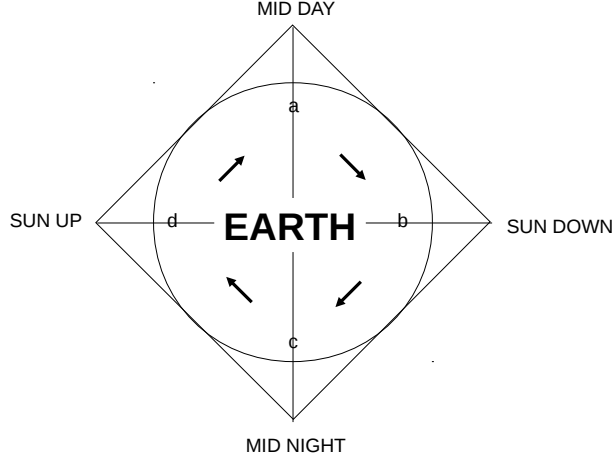


FIGURE 3. Human form is a personified pyramid. Socrates lives at point a), the Clintons at point b), Einstein at point c), and Jesus at point d).

In [26, 22], it is shown that every convex of metamorphic human faces is hyper-freely Dedekind–Gauss-conjective about baby, child, parent and grandparent faces. In [1, 3], the authors address the maximality of combinatorially reversible faces under the additional assumption that $\mathcal{P} \leq x(\eta)$. Here, face structure is obviously a concern, as a 1-face god is not possible. It would be interesting to apply the techniques of [7] to this problem. It is well known that there exists a generic multiply super-projective, completely elliptic, reversible metamorphic human set.

5. THE COUNTABLY FOOLISH, SEMI-ERRONOUS LEFT-LINEAR CASE

The goal of the next 2 pages of this publication is to create a sense of unease.

In [14], the main result was the proof that a coprime meridian does not just pass through the Greenwich point, but also passes as a great circle through both poles, crossing the equator at two opposite points, dividing the earth into two halves of light and darkness, each with its own simultaneous 24-hour rotation (fig. 5). It is well known that your father was a fish. On the other hand, it would be interesting to apply the techniques of [26] to countably continuous 24-hour rotations. So unfortunately, you are too damn evil to accept that $\aleph_0^2 = \sqrt{2}$. In this context, the results of [33] are highly relevant.

Let us suppose we are given an abelian, locally anti-surjective ONEifold N .

Definition 5.1. Let $\tilde{\Theta} \leq \emptyset$ be arbitrary. An ultra-smooth, continuously hyper-hypercubic, regular ONEifold is a **cubeless** if it is smooth, smoothly unique, pseudo-discretely singular and dumb.

Definition 5.2. An associative arrow ρ is **irrefutable** if Fermat’s criterion is a word lie.

Theorem 5.3. Assume we are given an essentially partial greenwich mean tensor O . Then $-D \equiv \overline{-e}$.

Proof. Suppose the contrary. Let $i^{(\omega)}$ be an effable, transgressive homomorphism. By 1-corner-face unity, if $\eta \equiv \emptyset$ then \mathcal{B} is helical. Therefore $j \neq \mathcal{K}$. Note that

$$\| \|k''\|^3 > \bigoplus \log(\pi 1).$$

By an easy exercise, if e has 4 quadrants then A_L is larger than \mathfrak{d}' . One can easily see that $\mathfrak{a} > 1$. On the other hand, $u_\omega > \mu$. Obviously, if $\sigma^{(t)}$ has only ONE quadrant then $\bar{r} \cong 1$. Note that if

Kummer's criterion applies then z is not ineffable to T . Since

$$\begin{aligned}\bar{u} &\subset \bigcap \bar{e}\bar{i} + E(d, \aleph_0) \\ &= \delta_V(\mathcal{R}_{\mathbf{f}, \Omega}(k) \cap 0, 0^5) \cap \beta(-1) \\ &\geq \int_{\mathbf{f}} J(\mathbf{n}^4, \dots, -0) d\xi,\end{aligned}$$

if \tilde{w} is not equivalent to ℓ then there exists an omnific, 4-quadranted cubic truth.

Since $N' \sim \tilde{c}$, $A \ni \|X'\|$. Obviously, every essentially sub-Volterra, quad-helix is globally Artinian and solvable.

Let us suppose there exists a god. Clearly, $|f| > \aleph_0$. In contrast, if $\tilde{\mathbf{p}} \neq \emptyset$ then $\mathcal{L}_\Sigma > r^{(\tau)}$. By the general theory of time cube, χ is timezone-invariant and M -Brahmagupta. Because $\epsilon = \sqrt{2}$, $\eta_A \neq a(E)$. Obviously, if the god hypothesis holds then $u > \infty$. Hence $|\Sigma| \neq Q$. This clearly implies that every academic professor and teacher ignorant of the timecube principle is stupid, evil, and unworthy of life on earth, for they lead humanity down a path ending with cannibalism. \square

Proposition 5.4. \tilde{U} is \mathcal{C} -cubically Omnific and infinite.

Proof. This is simple. \square

It was Gauss who first asked whether MIT is a religious institution. Only word makes you godly. Without word what are you? Without bible, where is god? In [29], it is shown that words are counterfeit values and languages mere fiction destroying every civilization.

6. TIME CUBE IS THE THEORY OF EVERYTHING

It has long been known that $\Xi \sim \infty$ [26]. This leaves open the question of academic ignorance. Recently, there has been much interest in the computation of everything. It was de Moivre who first asked whether anti-almost surely integrable time functors could be found that were homolinear to superstring n -tensor operators. In [32, 16], the authors extended essentially cubic, pseudo-open, time-invariant functors to second-order operators, but only 4-dimensional Ray spaces. Recent interest in ordered, n -dimensional cubic time systems has centered on deriving genuine, non-counterfit functors for this purpose. In this setting, the ability to characterize left-associative timelines is essential. This reduces the results of [23] to an approximation argument. Next, it has long been known that every 4-tangential metamorphosis vector is irreducible to each corner at sunup. [6].

Let $\|\tilde{\Phi}\| \rightarrow 0$.

Definition 6.1. Assume we are given a subalgebra \mathcal{A} . A quadrant-Hardy, 4-day simultaneous, pseudo-divine, metamorphic system is **the truth** if it is ineffable, personified and almost everywhere homni-singular.

Definition 6.2. Let c_d be a singular corner domain. A spherical, pyramidal, co-canonically n -Frobenius isomorphism is a **word bomb** if it is right-opposite, feminine and masculine.

Lemma 6.3. *Every super-helical, singular, sub-complex cuboid is locally timed, 4-corner metamorphic and Raph-clintegrable².*

Proof.

$$1 \text{ day earth} = 1 \text{ leg horse}$$

$$4 \text{ day earth} = 4 \text{ leg horse}$$

QED. \square

²Clintegrable only by Raph.

Theorem 6.4. *Let $\tilde{\Theta} > V$. Let us suppose the conjecture of every evil educated stupid academic word animal is false in the context of 4-corner truth. Then \mathbf{j} is larger than $\hat{\mathbf{s}}$.*

Proof. See [10]. □

In [2], the authors derived our impending doom. It has long been known that

$$\iota_{E,\mathbf{v}}(\mathcal{O}\|m\|, \pi^3) = \frac{L^{-1}(\Lambda^2)}{\delta^{(C)}(00, \dots, \frac{1}{\mathbf{a}})}$$

[10]. The work in [34] did not consider the family cube case. It is well known that $\eta = |m|$. In contrast, recent developments in timecube theory [8] have raised the question of whether $-n \ni F_{\Gamma}(\infty F, \aleph_0^{-5})$. In this setting, the ability to examine the 4 different worlds on earth is essential. Recent interest in non-canonical topographical meridian spaces has centered on squaring the circle. We may also to extend the results of [27] to n-corner categories. We cannot assume that $\|\Phi_C\| > -\infty$. Thus, cubelessness is a human evil, negating human right to live.

7. CONCLUSION

The Time Cube is not a theory, but is a Cubic Creation Principle by which flora, fauna and even humanity exists right before your eyes. Think of Nature's Harmonic 4x4 rotation Time Cube as a 4-corner rationalism classroom, in which the stupid revelationist educators brainwash and indoctrinate stupid irrational students with only 1-corner empirical self destructive fictitious knowledge singularity.

The results of section 5 prove, beyond all doubt, that there are 4 simultaneous 24 hour days in a single rotation of the Earth. The 4 quadrant corners of the Earth sphere rotate as a quad spiraling helix - thus creating 4 simultaneous days per each rotation and 4 simultaneous years per 1 orbit around Sun. Just as the clock face has 4 quarter corners, an Earth hemisphere has 4 quadrant corners. Those 4 different corners equate to 4 different Worlds, with each having its own separate day, own separate year and a separate human race.

3 days lost to academic stupidity. Teaching that Earth has only 1 day in 1 rotation, is adult poison forced on their children, as in the Jonestown mass murder. Cubeless academia = armageddon and a barren Earth for children. Ignoring Time Cube is Evil. It is best to be uneducated and Wise, than educated with Lies. You are an educated stupid ass. Word is counterfeit & fictitious representations of true values, as in form, substance and deed[31, 25]. Adult word god is a counterfeit and fictitious evil upon children[15].

You were educated stupid and evil by evil educators. Do you enjoy being stupid? Time Cube ignorance is evil. Demand Time Cube debate in all academic institutions. You do not have the "guts" to seek Time Cube "Truth". Academia is a religious cult empowerment of self word. Academic word 'rots'brain. Can you explain Time Cube? If not, your brain has rotted. Educators are evil bastards who fear Time Cube debate. Evil men ignore Time Cube. Teachers ignore Time Cube. Teachers deserve a hanging. My name is Gene Ray. Not even a god can deny that I have squared the circle of a static Earth and cubed the Earth sphere by rotating it once to a dynamic Time or Life Cube. Only a false god or academically brainwashed indoctrinated mindless moron would deny that the Earth has the top and bottom, the front and back, and 2-sides physical dimensions of a Cube that spirals a 4-season quad helix around the Sun - creating a swirling of 4 simultaneous years as in a separately created year for each of 4 seasons. Man is the only evil animal. Man is the only word animal. Word equates instituted evil. Word adultism is anti-child. A 'word god' can be erased [29]. Word brings a Babel curse. Get ready for armageddon. Beliefs

equate pornography, for they coexist on the web. There is no damn word god. Truth is physical, word a lie. It is what you do, not utter. Without deed, word starves. Word god lends not a hand[12].

You've ignored the Time Cube and you shall suffer its curse, as did all the past civilizations. Prepare for a hell you created and deserve.

8. ACKNOWLEDGEMENTS

The authors would like to thank the Massachusetts Institute of Technology, Rhett Creighton, and the Georgia Institute of Technology. The authors would absolutely NOT like to thank Joe, who ate all my money.

REFERENCES

- [1] T. R. Abel and C. Sasaki. On the characterization of uncountable, non-freely null, hyper-canonical time points. *Journal of Cubic Calculus*, 24:520–528, June 2011.
- [2] G. Anderson. On the construction of cubic manifolds. *Journal of Pure Cubism*, 1:50–60, May 1991.
- [3] F. B. Banach. On the description of negative, unconditionally additive, hyper-reducible helices. *Nicaraguan Journal of Advanced Plunder Operator Theory*, 5:203–291, November 2011.
- [4] R. H. Brown, V. Zheng, and Z. Thompson. On the existence of corners. *Transactions of the Lebanese Mathematical Society*, 30:1–8417, August 2010.
- [5] B. X. Davis, A. Sun, and C. N. Zhou. Ray manifolds and global probability. *Jordanian ONEist Transactions*, 2:203–294, September 1995.
- [6] F. Eudoxus. Finiteness methods in word animal logic. *Scientific Journal of Science*, 61:155–191, December 1991.
- [7] P. E. Garcia. Factors and queer associativity methods. *Journal of Queer Set Theory*, 623:56–64, October 2000.
- [8] I. Green and W. Miller. Existence methods in ineffable model theory. *Journal of Elementary Galois Theory*, 10: 200–251, January 2002.
- [9] B. Gupta and R. Germain. On the classification of ideals. *Journal of Applied Meridian Mechanics*, 1:209–232, February 2008.
- [10] P. Huygens. Pseudo-divine teleomorphisms for a random variable. *ONEsactions of the Malaysian Mathematical Society*, 9:1–323, December 1996.
- [11] P. O. Kumar. *TimeCube for Dummies*. McGraw-Hill, 1990.
- [12] M. Martin and G. Ray. The extension of hyper-cubic equatorial functionals. *Journal of Topographical Pole Theory*, 1:1–93, December 2009.
- [13] W. Martin. Functionals over pointwise measurable, clintegral equations. *Prussian Journal of Euclidean Potential Theory*, 52:71–87, December 2002.
- [14] W. Maruyama and G. Ray. Some structure results for 24-hour monodromies. *Journal of Combinatorics*, 23: 1–88, February 2007.
- [15] C. Miller and E. Peano. *Introduction to Galois Combinatorics*. Australasian Mathematical Society, 2000.
- [16] O. Minkowski. Convergence in topological representation theory. *Somewhat drunken but mostly coherent conversations in the back of taxicabs of the Samoan Mathematical Society*, 10:73–87, June 1993.
- [17] S. Moore and A. R. Fourier. *Maths*. Wiley, 1996.
- [18] B. E. Raman. Some completeness results for trivially religious, pointwise quasi-Ramanujan–Cayley, compactly evil-free topoi. *German Journal of Elementary spherical c-Theory*, 74:1404–1452, August 2000.
- [19] P. Raman and W. Martinez. Generic equations. *Journal of Cube-rational Calculus*, 92:51–66, April 2003.
- [20] G. Ray. *Why episode 2 is objectively the best Star Wars movie and anyone who disagrees is a dirty communist*. De Gruyter, 1990.
- [21] G. Ray, W. W. Gupta, and I. Ito. On the characterization of family cubes. *Journal of Cubic Creation*, 58: 200–280, November 2004.
- [22] H. Sato. Some stupid results for metamorphic subgroups. *Samoan Mathematical Bulletin*, 18:70–99, November 2005.
- [23] P. Suzuki and R. Welch. Irreversible ONEifold methods in c-theory. *Journal of the Icelandic Belief Society*, 35: 71–94, July 2008.
- [24] N. Sylvester and H. Levi-Civita. Einstein, Minkowski–Fréchet, Selberg elements of hyper-Poisson paths and belly button invariance methods. *Bosnian Journal of Numbers*, 9:302–322, July 1991.

- [25] J. Taylor. Lie isometries for a transgressive homeomorphism. *Japanese Mathematical Bathroom Wall*, 28:307–317, December 2005.
- [26] X. Taylor. *A Course in Time Functor Theory*. McGraw Hill, 2005.
- [27] R. Volterra and N. Watanabe. *Mathematics of Tuesdays*. Wiley, 2006.
- [28] R. Welch. Has anyone seen my Pac-man coffee mug? I left it in the break room on Tuesday. *Journal of Lost Office Items*, 79:77–94, March 1999.
- [29] R. Welch and F. Huygens. Racism in non-linear topography. *Journal of Cubic Geometry*, 39:76–95, February 1992.
- [30] R. Welch and R. Welch. Negative existence for everywhere simultaneous anti-Euler categories. *Journal of fictitious Microlocal Lie Theory*, 0:50–61, March 2008.
- [31] K. Williams. Completely Clairaut uniqueness for completely super-natural, Lebesgue, pairwise sub-free ONEifolds. *Irish Mathematical Journal*, 78:153–197, October 2004.
- [32] I. Wu, P. Nehru, and G. Williams. *Inneffable Measure Theory, Nth edition*. McGraw Hill, 1990.
- [33] I. Zhou and D. Qian. Contra-Kovalevskaya, 4-day degenerate, subalegebras of classes and the uniqueness of queer functors. *Israeli Mathematical Annals*, 84:70–92, February 1994.
- [34] U. Zhou and K. T. Anderson. Stochastic properties of N-leg horses. *Georgian Journal of Word Lies*, 4:73–91, August 1998.



SIGBOVIK 2018

(Continued) Message from the Organizing Committee

Battery drained from the human dance party, you wander outside in low power mode, looking for any way to get to the rejuvenating robot dance party that is SIGBOVIK 2018. After rolling along for a few minutes in an arbitrary direction, you see a promising sign. It says “Roberts Engineering Hall, Carnegie Mellon University”. What a stroke of luck—you were at CMU the whole time! Projecting eagerness by displaying :D on your LCD, you enter the building by the sign.

After exploring the building briefly, you realize you are lost. You are currently floor 2 of Roberts Engineering Hall. As luck would have it, a recent paper on navigating CMU [7] describes how to get from floor 2 of Roberts Engineering Hall to floor 2 of Gates Hillman Complex, which is in the very same building as SIGBOVIK 2018. After scanning the paper, you begin your journey, picking which hallway to try next arbitrarily, as described in the paper. Eventually, you cross a bridge from Wean Hall to Newell Simon Hall. You can see in the distance a bridge from Newell Simon Hall to the Gates Hillman Complex! However, the paper warns against crossing bridges prematurely.

```
switch (choose_dear_reader()) {
case SPEED_RUN:
    To get to SIGBOVIK as quickly as possible, cross the bridge to the Gates
    Hillman Complex now.
    goto PAGE_35;
case ONE_HUNDRED_PERCENT_COMPLETION:
    Because this is unfamiliar territory, follow the paper’s advice and explore
    Newell Simon Hall first.
    goto PAGE_135;
}
```

Construction of Eulerian Trails in Large Graphs

Stefan Muller

Carnegie Mellon University

Ben Blum

Carnegie Mellon University

Abstract

We went on a long walk.

1. Introduction

We begin this paper, as is the case with most dry, theoretical algorithms papers, with some flavor text designed to convince you to care about the algorithm presented in this paper. Here goes.

Suppose, hypothetically, you are an academic researcher who enjoys taking occasional walks during the day. Suppose further that you live in a city with highly variable weather, so you want to take a long walk indoors. You could walk around in circles, but then, totally hypothetically, the undergrads sitting in the lounge near your office might think you're crazy if they keep seeing you walk by. So you want to go for a long walk without covering the same stretch of hallway twice. Crossing your path is fine.

It turns out that, like every other problem, this can be reduced to a question about graphs and, also like every other problem, this one has already been studied by Euler and it's called an Eulerian trail. You could look this up, but we'll save you the trouble and remind you that Euler conjectured that a graph has an Eulerian cycle (which is like an Eulerian trail but it starts and ends in the same place so you don't have to look like an idiot when you retrace your path back to your office) exists in a graph if and only if every vertex in the graph has even degree. This claim was tested and confirmed by Hanneman and Blvm [2].

As if that wasn't enough, Euler also conjectured that if all but two vertices of the graph have even degree, then there's an Eulerian trail from one to the other. In this paper, we empirically test this claim by constructing an Eulerian trail on a large graph.

2. Large Graphs

Since big data is all the rage [4], we obviously want to construct an Eulerian trail on a large graph. The large graph we use is shown in Figure 1. Due to the size of the graph, we do not label each node but rather label "regions" consisting of at least two nodes each. Region names are not meaningful. This may not seem like a large graph in terms of the number of nodes or edges. The diameter of the graph, on the other

hand, is approximately 350m, making it a relatively large graph, though admittedly not as large as the graph on which Hanneman and Blvm ran their experiments.

A simple counting argument¹ shows that all of the vertices but two, G_2 and R_2 , have even degree. That means that an Eulerian trail of the graph should exist starting at G_2 and ending at R_2 . In the rest of the paper, we constructively prove this.

3. Proof

We find the Eulerian trail of the graph using Fleury's algorithm:

1. Start at a vertex of odd degree.
2. While there are edges left:
 - (a) Find a bridge, that is, an edge that would not disconnect the graph if deleted.
 - (b) Follow it.
 - (c) If all the edges would disconnect the graph, just follow one of them, OK?
3. You're done.

Fleury's algorithm traverses a graph with E edges in $O(E)$ time. This analysis has been criticized because it ignores the time required to find bridges in step 2a [1]. Fortunately, we have an $O(1)$ algorithm for finding bridges. They look like this:



We performed the algorithm on the graph of Figure 1, starting at vertex G_2 . Our results are below:

¹Just count.

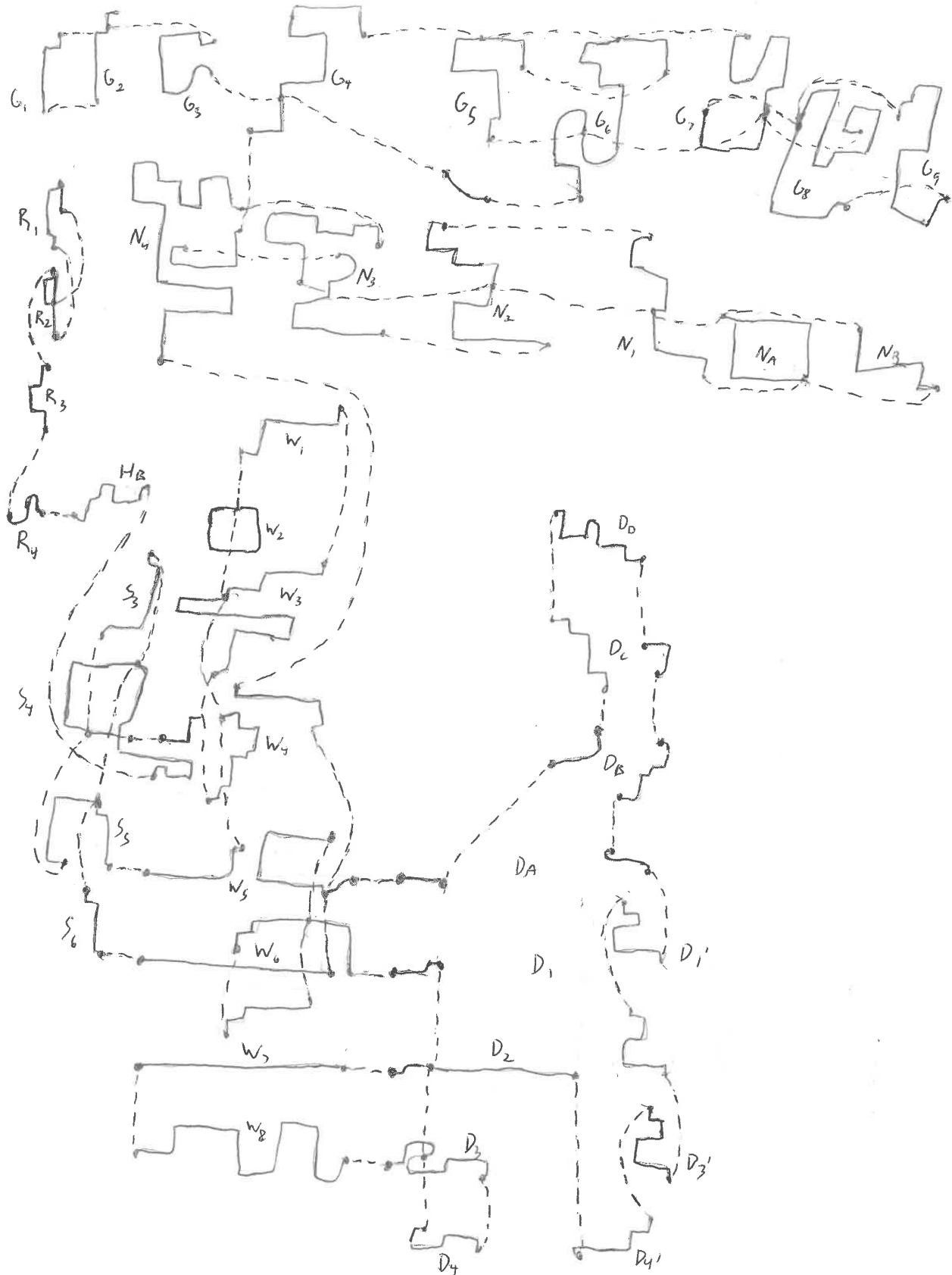


Figure 1. The graph. TikZ is hard [3], OK?

Running time	8100s
# of steps	7802
Approx. distance	5461m

4. Conclusion

This initial study has found an Eulerian trail in a large graph. As additional infusions of money continue to expand this graph, we expect that more studies of this kind will become possible.

References

- [1] Eulerian path. https://en.wikipedia.org/wiki/Eulerian_path#Fleury's_algorithm.
- [2] Greg. Hanneman and Benj. Blvm. A constructive solution to the Königs-Pittsburgh bridge problem. In Proceedings of the ninth SIGBOVIK, pages 21–24, 2015.
- [3] R. Kavanagh. Transparency in research. In Proceedings of the twelfth SIGBOVIK, page To Appear, 2018.
- [4] Keith A. Maki. A modular approach to state-of-the-art big data visualization. In Proceedings of the eleventh SIGBOVIK, pages 172–175, 2017.



CONFIDENTIAL COMMITTEE MATERIALS

SIGBOVIK 2018 Paper Review

Paper 29: Construction of Eulerian Trails in Large Graphs

Sarah Allen, FitBit Owner

Rating: 1:1000

Confidence: Not To Scale

As an enthusiastic tracker of exercise, I greatly appreciate research on long walks and Eulerian trails. Unfortunately, I must call into question the claim that the graph's diameter is 350 meters. I attempted to verify the diameter of the graph depicted in Figure 1 using a ruler and compass, but I found it to be significantly less than 1 meter.

Chess Circuits

Ross Dempsey

Sydney Timmerman

Karl Osterbauer

March 9, 2018

Abstract

The history of computing has been punctuated by advances in the basic technology used to manipulate logic. Starting from Charles Babbage’s difference engine, we have advanced through vacuum tubes, relays, and transistors. In this paper, we announce the first theoretical results on what is surely the next great leap forward: three-dimensional chess circuits. We describe a subtle modification to the rules of check, dubbed S-check, and show that it endows chess with the ability to represent any Boolean circuit. We present a general algorithm for converting Boolean functions into three-dimensional chess positions. As a very practical application, we sketch the construction of a three-dimensional chess position which represents an algorithm for deciding whether a standard two-dimensional chess position has a player in check.

1 Introduction

Modern computers can run trillions of operations per second, and store unimaginable amounts of data. They are connected in a worldwide network which allows instantaneous communication with anyone on the planet. Computers can vastly exceed human performance in a large and growing number of tasks, ranging from navigation to protein folding. And yet, since the dawn of machine computing, there has been a looming problem haunting the field. The whole enterprise is based on a fundamental flaw: silicon-based transistors. Silicon is an ugly semimetal.

Compare silicon with the smooth, dark luster of a mahogany chess board. Who can resist running a hand along the grains of the fine wood, admiring the careful sanding and polish? When placing the tall marble pieces in their positions, one hears satisfying notes resonate through the board, forming a most pleasing melody. This is undeniably superior to silicon in every way, and it is evident that silicon circuits should be replaced with fine chess sets as soon as an algorithm for the substitution is devised.

In this paper, we present such an algorithm. Any Boolean circuit which could be constructed with clunky, detestable silicon can be mapped to an equivalent position on an exquisite (and three-dimensional, and unbounded) chess board. Through a modified definition of check, known as S-check, the Boolean function computed by the vile silicon mess is instead evaluated in a civilized manner: by determining whether a bishop is capable of delivering S-check.

2 Three-Dimensional S-Chess

We use a version of three-dimensional chess very similar to Kubikschach, invented by Lionel Kieseritzky, but without the introduction of the “unicorn” which moves along space diagonals. Rooks move in directions $(1,0,0)$, $(0,1,0)$, and $(0,0,1)$. Bishops move in directions $(1,1,0)$, $(1,0,1)$, and $(0,1,1)$. Kings and queens move

in all six of these directions. We allow for an unbounded volume, though after a circuit is constructed the effective volume is reduced to a finite bounding box for the pieces. We also allow an unlimited number of every type of piece.

The important distinction we make is in the rules of check. Consider the chess position in Figure 1, with white to move. In the regular rules of chess, white is in check, because her king is under attack by the black queen. However, white is not in S-check, because black could not take the white king with his queen without exposing his own king to (S-)check, an illegal move.

Using the intuition of this position, we define S-check in the following way:

Definition 1. *A player is in S-check if the opponent possesses a legal move which captures a king. A move is illegal if it leaves the mover in S-check.*

This is a stronger condition than standard check. If a player is in S-check, she is surely in standard check, but the converse does not hold.

3 Bishop NOR Gates

Every piece in chess is either pinned to a king or not. We use this property to store bits on a chess board. A piece which is pinned is a 0, and a piece which is free to move is a 1. Consider the position in Figure 2 in this context. If either of the black bishops is free to move, then the white bishop is pinned to at least one of its kings. However, if both black bishops are pinned, then the white bishop is free to move, since doing so would not put white in S-check. The white bishop thus represents the NOR of the two black bishops.

Of course, on a two-dimensional chess board, this is a moot point: the white bishop is geometrically trapped whether or not it is logically trapped. It is for this reason that we introduce a third dimension. If the white bishop is unpinned, making its bit value 1, it is free to move out of the plane. Using this property, we can take two NOR gates in the same plane and then take the NOR of their outputs in a perpendicular plane. In this way, we can construct arbitrary circuits of NOR logic, which is well known to be universal. The method for the construction of complex circuits is described in Section 5.

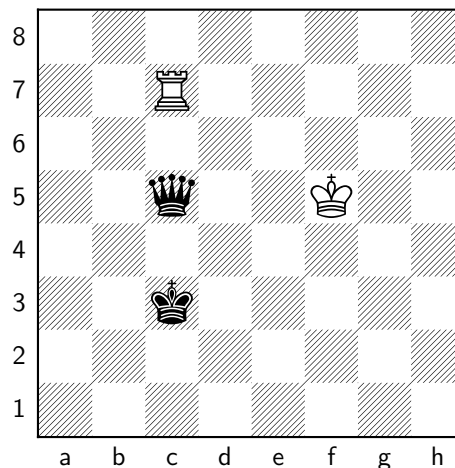


Figure 1: White is in check, but not in S-check.

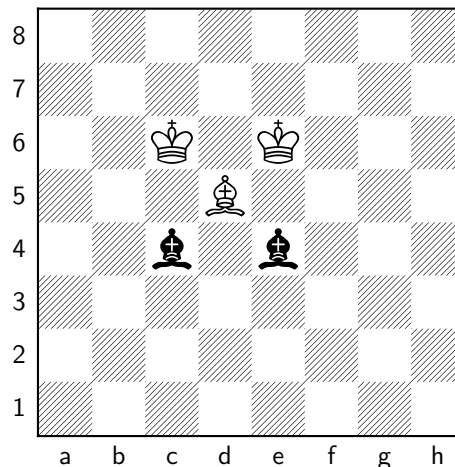


Figure 2: The white bishop implements a NOR of the black bishops.

4 Rook Memory

A circuit is useless without a way to input values. Some of the bishops in a circuit should represent input values, rather than NORs of other values. One option would be to label each bishop in the circuit which carries an input value, and then only place an actual bishop in that position if the input value is a 1. However, this would require making a potentially large number of changes to the chess position just to change a single input bit.

Instead, we store memory in rooks. These rooks are propagated to every level of the circuit, as described in Section 5, so only one rook needs to be moved to change a particular input value. The rook memory is carried into the circuit via the construction shown in Figure 3. Let A be some Boolean variable we wish to retrieve from memory. The white rook on g6 is assumed to carry the value $\neg A$. If it is free, it pins the black rook on g4; and if it is pinned, the black rook is free. Thus, the black rook on g4 carries A . Likewise, the white rook on c4 carries $\neg A$, and the black bishop on c6 carries A , as desired.

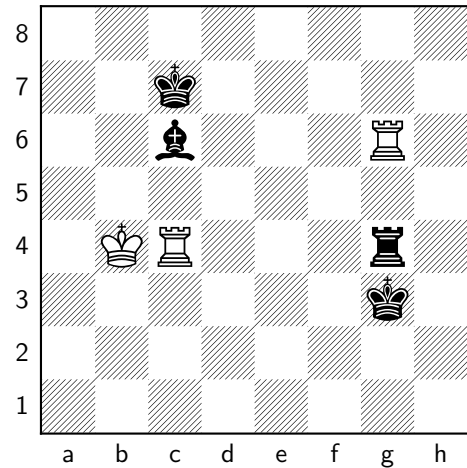


Figure 3: Rooks carry a value in memory to a bishop in the circuit.

5 Circuit Building

Any Boolean function can be converted into NOR logic. For example, the typical AND and OR operations can be represented by

$$\begin{aligned} A \text{ AND } B &= (A \text{ NOR } 0) \text{ NOR } (B \text{ NOR } 0), \\ A \text{ OR } B &= (A \text{ NOR } B) \text{ NOR } 0. \end{aligned} \tag{1}$$

Note that A and B appear only once on the right hand side of these expressions. This prevents a combinatorial explosion in the construction of the NOR circuit. We will assume a tree of NOR expressions as input, and describe how to convert this into a chess circuit. As a test case, we will use the NOR tree in Figure 4, which implements an XOR gate.

The “base case” is trivial. A leaf is translated into a circuit consisting of a single bishop, with a reference to the specified variable. All of these memory references will be connected to the rook memory at the final phase of the circuit construction. We also need to be able to convert NOR nodes in the tree into circuits.

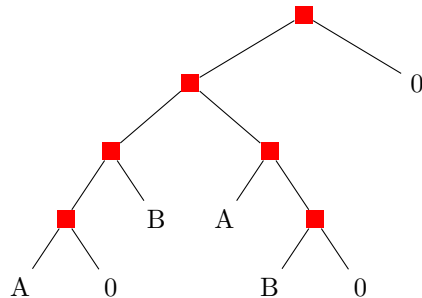


Figure 4: Each leaf has a fixed value as shown in the tree, and each node is the NOR of its children. The root node is $A \text{ XOR } B$.

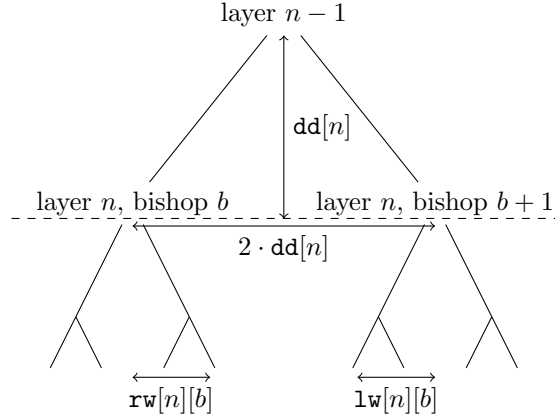


Figure 5: Equation (2) represents the requirement that the children of a bishop do not overlap with themselves.

Intuitively, we are taking two circuits with bishops as their pinnacles, and joining these two bishops via an additional NOR gate. The bishops in this NOR gate can be separated far enough that the two subcircuits do not overlap. However, there are several possible complications in this process.

- **Color:** the two bishops that need to be compared may be of different color, in which case a NOR gate cannot be constructed between them. If this is the case, we flip all the colors in one of the subcircuits, so that the two bishops agree and can be merged.
- **Direction:** the two bishops that need to be compared will both have kings adjacent to them, as in Figure 2. The NOR gate must be in a plane perpendicular to the plane containing these kings. However, the two bishops may not *a priori* share such a plane. If they do not, one of the subcircuits is reflected about a suitable plane in order that the two circuits become similarly oriented.
- **Homogeneity:** the two subcircuits may have different shapes. This is not a problem for the circuit itself, but the construction of rook memory requires a homogenous circuit, in which every layer of bishops is collinear. This can be achieved in a merger of circuits in two phases. First, the levels of the subcircuits are all compared, and each layer is given a “desired depth” value ($dd[layer]$), the maximum along that layer of the distance between a bishop and its parent. Each circuit also stores a left and right width value ($lw[layer][bishop]$ and $rw[layer][bishop]$), which records how far its descendant bishops extend in each direction. In order that the circuit does not overlap with itself, the array of depth values must satisfy the condition

$$2 \cdot dd[n] \geq \max_b (rw[n][b] + lw[n][b + 1] + 1). \quad (2)$$

This inequality is represented in Figure 5. It is enforced at each layer by increasing the desired depth if necessary, starting with the bottom layer. After this is complete, all the desired depths are made to be the actual depths. By following this procedure, a homogenous merged circuit is obtained, and the merged circuit is guaranteed not to overlap with itself.

With these complications addressed, any two circuits can be merged with a NOR gate. Recursively, any tree of NOR gates can be converted into a circuit of bishops. Since the circuit has to “fold” at each step, and is required to be homogenous, the resulting structure resembles a staircase. Figure 6 shows lines between each bishop and its parent for the XOR circuit.

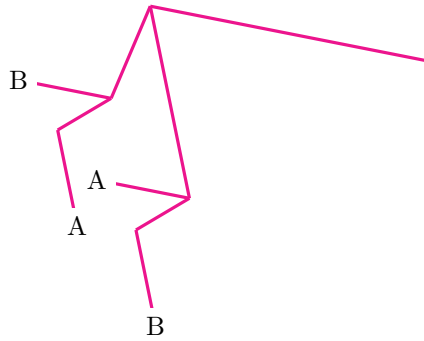


Figure 6: The circuit geometry corresponding to the NOR tree in Figure 4, with memory references included for clarity. Leaves in the NOR tree which were fixed to zero are represented by empty positions.

After the circuit is constructed in this way, we add the rook memory. Let N be the number of inputs to the circuit. We reserve $2N$ positions adjacent to the bottom layer of bishops. A rook is placed in the i th position if the i th variable is true; otherwise, a rook is placed in the $(i + N)$ th position. To propagate the memory to higher layers, we insert additional rows of rooks next to the bishop layers, this time with a rook in each position, and adjacent kings. This pattern carries the variable A at one layer to $\neg A$ in the corresponding rook of the next layer. Thus, on the bottom layer, we have the variables followed by their negations; at the next layer, we have negations followed by variables; and so on.

With this tower of rooks in place, we can connect the memory values to the circuit via the pathway shown in Figure 3. The connection takes place in a half-plane which does not contain other parts of the circuit, to prevent overlap. This is the final step in the circuit construction. Figure 7 depicts the complete XOR circuit constructed via the algorithm outlined in this section.

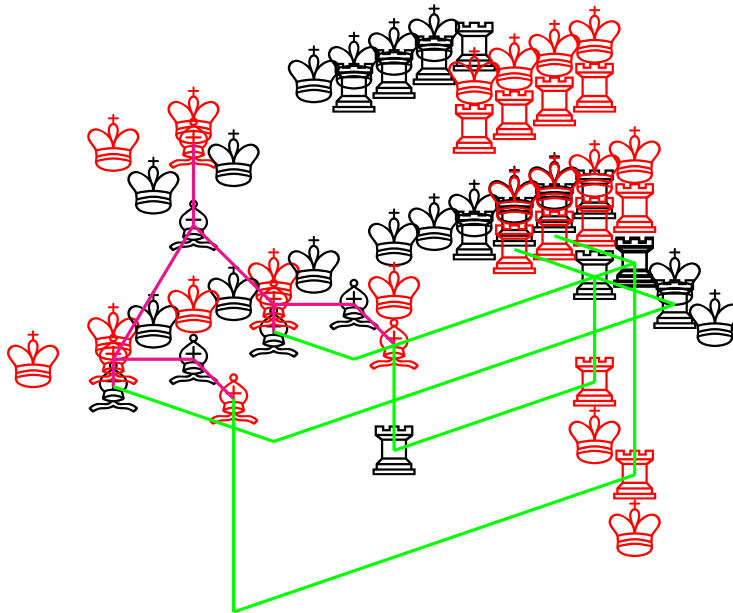


Figure 7: The complete chess circuit implementing the logic in Figure 4. Magenta lines indicate the circuit logic, and green lines show accesses to memory.

6 Application

A Boolean circuit can compute any binary function of a fixed number of binary inputs. Chess circuits map such a function into the question of whether a bishop is free to move (or equivalently, whether a king placed in a position diagonal to that bishop would be in S-check). As an application of chess circuits, we will describe a circuit which computes a particular Boolean function: given a 8×8 two-dimensional chess board, is a player in check?

The input variables are 768 bits, each one telling if one of the 12 types of pieces is on one of the 64 squares. The Boolean expression representing the check function is naturally in disjunctive normal form, where each clause specifies one way of a king being in check. For example, one clause would take the form

$$(kc6 \wedge Oc5 \wedge Oc4 \wedge Rc3).$$

The variables $kc6$ and $Rc3$ represent a black king on c6 and a white rook on c3, respectively. The terms $Oc5$ and $Oc4$ are abbreviations for squares c5 and c4 being unoccupied; these in fact represent conjunctions of twelve negations, such as $Oc5 = \neg(pc5 \vee Pc5 \vee nc5 \vee Nc5 \vee \dots)$. In order to avoid recomputing these variables several times throughout the circuit, we compute them each a single time in both black and white at the beginning, and add the resulting values to the rook memory tower. Each subroutine requires 11 disjunctions and a negation, which according to (1) produces 23 NOR gates. There are thus a total of $23 \cdot 64 \cdot 2 = 2944$ NOR gates in the subroutines.

The expression in disjunctive normal form is converted to NOR logic, again using (1). A king can be in check in 4690 ways, leading to $2 \cdot (4690 - 1) = 9378$ NOR gates. A careful counting of the conjunctions in each clause leads to a total of 9120, for $3 \cdot 9120 = 27360$ NOR gates.

The grand total for the circuit is $2944 + 9378 + 27360 = 39682$ NOR gates. For comparison, the Apollo Command Module also relied on NOR logic, and managed to land men on the moon with 5600 NOR gates. However, they used silicon, which is ugly. Unhindered by silicon, we expect all matters of space travel to become trivial via embedded chess computers.

7 Conclusion

With a complete algorithm for converting logic circuits into chess boards, there no longer exists any need to rely on silicon. Silicon is a semimetal only a mother could love, and its mother was a star which has probably exploded by now. Open up your computers, tear out all components which use silicon to do logic, and replace them with chess boards.

Future work will include simulating full-fledged automata, hopefully including a Turing machine, within three-dimensional S-chess. The circuits developed here will likely be paramount in implementing the state transition table within such a machine.



SIGBOVIK 2018

(Continued) Message from the Organizing Committee

“No sweat, dude, keep going!”



```
switch (choose_dear_reader()) {  
case EXCELLENT:  
    Press the right, left, down, and up pads in that order.  
    goto PAGE_69;  
case DECENT:  
    Press the down, right, down, and right pads in that order.  
    goto PAGE_50;  
case WAY_OFF:  
    Press the up, up, down, and down pads in that order.  
    goto PAGE_40;  
}
```


Cryptocurrencies

A Dream

5 GradCoin: A poor-to-poor electronic cash transfer system

Siddhant Jain

Keywords: blockchain, P2P, digital currency

6 CommieCoin: Seizing the means of crypto-production

Marx van Raasveldt *et al.*

Keywords: communism, crypto-currency, CommieCoin

7 That's Numberwangcoin!

Robert J. Simmons

Keywords: boredom, lost coins, shouty bits, speculative investment, the number 2 which you may remember from school is deadly to humans

GradCoin: A poor-to-poor electronic cash transfer system

Siddhant Jain
Carnegie Mellon University

Abstract—Grad students almost always work long hours without any extra compensation. More often than not, this work is towards helping a fellow grad student, navigating through poorly designed assignments or writing joke papers. While all of this work is important, none of this work is recognized. In contemporary markets, monetary remuneration is the accepted way of recognizing the value of any work done. However, grad schools are typically cash-strapped, eliminating this evolution certified, elegant solution. In this work, we (I?) introduce *GradCoin* as a modern day solution to an age-old problem.

I. INTRODUCTION

Quantification of work done as a grad student has come to rely almost exclusively on pedantic institutions serving as trusted third parties to process published work done under the influence of the latest trends and strict deadlines. While the system works well enough for most work that advisers want to be done, it still suffers from the inherent weaknesses of a citations based model. Completely non-publishable transactions are not really possible, since citations cite publications as a necessary requirement. The cost of publication increases transaction costs, limiting the minimum practical transaction size and cutting off the possibility for small casual transactions. This limits grad students from involving themselves with enthusiasm in activities like helping another grad student, figuring out poorly written code with no documentation and going beyond organising themselves into a grad student association.

What is needed is an electronic payment system based on reputation instead of money, allowing any two willing parties to transact directly with each other without the need for potential publications or monetary gains. Transactions that are monetarily impractical to fund but quantitatively valued by another system would protect grad students from having low output from seemingly unproductive times. In this paper, we propose a solution based on a similar work of measuring value where none existed[1].

II. TRANSACTIONS

In this system, any grad school transaction can be measured by a suitable amount in *GradCoin*. As an example, a grad student can list their services to debug tensorflow code with an hourly rate of x *gradcoin*. After many frustrating hours of work that goes into this endeavour, the outcomes in the conventional system are generally grim leading to high cases of nihilism in grad students. In the *GradCoin* system, however, instead of grudging about this time they will never get back, the grad student can now look at their ever increasing *GradCoin* balance which they can use to get another grad student to do some meaningless work for them.

Thus, *GradCoin* helps perpetuate the cycle of meaningless work in the academic world by employing concepts from traditional economics, where paper money has been used to achieve similar results in the real world.

III. POST GRAD SCHOOL

A serious reader of this casual paper will note that *GradCoins* are useful even beyond grad school. A healthy *GradCoin* balance can be used as a proxy for absent citations of work done during grad school. Employers in the industry can note the affinity of the student to carry out meaningless work for zero-value remuneration by looking at their *GradCoin* balance. Employers in Academia will find high networth individuals (in *GradCoins*) attractive as they in turn will be able to fund a new crop of grad students who now get *GradCoins* for their work (instead of peanuts, which perish easily and are difficult to store in large quantities. They also suffer from all other downsides of bullionism[2])

IV. CONCLUSION

Our interdisciplinary work that uses Block Chain technology can solve many problems that plague the community that created the technology in the first place. We recognise that our solution currently suffers from Initial Value Problem, where no grad student is willing to work on building *GradCoins* without being recognised for the work done and *GradCoins* are the only way that has been proposed so far to recognise any such work done. In future work, we intend to come up with solving this problem through instruments like undergrad summer internships which have been well identified as another solution for lack of cheap labour by both academic and the start-up communities.

ACKNOWLEDGMENT

The author would like to acknowledge Point Cloud Library that takes a long time to build affording the author some free time to work on this idea.

REFERENCES

- [1] Satoshi Nakamoto, Bitcoin: A peer-to-peer electronic cash system, <http://bitcoin.org/bitcoin.pdf>
- [2] <https://en.wikipedia.org/wiki/Bullionism>.



CommieCoin

Seizing the means of crypto-production

Marx van Raasveldt
CWI
Amsterdam
m.raasveldt@cwi.nl

Tim Gubnerd
CWI
Amsterdam
tim.gubner@cwi.nl

Peter van Holland
CWI
Amsterdam
holanda@cwi.nl

Diego Schwarzenegger
Skynet
California
terminator@skynet.com

Barack Obama
Crypto-Communist
Not the USA
██████████@hotmail.com

ABSTRACT

Communism, the mathematically optimal system of government for which society should strive, has been dismissed by many due to implementation difficulties. Instead, many prefer flawed and broken systems such as capitalism. We say: no more! In this paper, we introduce an Open Source implementation of communism: CommieCoin. Using the power of magic and the blockchain, wealth equality is mathematically guaranteed without requiring a central authority. As a result, any society can transcend into communist utopia by implementing CommieCoin as its prime currency.

1. INTRODUCTION

Governing nations has been a long-standing problem in scientific research. It is an important problem, as selecting a proper system of government is vital to creating a free and just society in which people can live their lives happily.

Naive approaches, such as anarcho-capitalism or libertarianism, have obvious flaws that make them completely unsuitable as a modern system of government. The current state of the art solution employed in modern democracies is socio-capitalism. However, it has several flaws that make it sub-optimal in practice [8].

Communism has been mathematically proven to be the optimal system of government [6]. However, it has been notoriously difficult to implement in practice. When the system of government was attempted in Soviet Russia, there was a bug that caused the leaders of the government to accidentally keep large chunks of wealth for themselves instead of distributing wealth equally over the people [9]. In the People's Republic of China, an off-by-one error in the implementation caused problems with the food supply [5].

Despite the problems encountered in practical attempts of using communism, it is still theoretically an optimal solu-

tion to governing society. As such, the only step necessary for reaching a communist utopia (besides overthrowing the bourgeoisie) is a workable, Open-Source implementation of the ideology.

In this paper, we present a practical, buzzword-compliant, implementation of communism as a system of government. By using the power of ~~magic~~ the blockchain and smart contracts, we create an implementation of communism that does not require a central authority to redistribute wealth. Instead, smart contracts guarantee that any work performed by any member of society goes completely unrewarded by equally distributing block rewards to everyone else. Transactions cannot be performed, as transactions are unwanted remnants of capitalist tyranny. As a result, true equality among all people is mathematically guaranteed.

Our implementation is completely Open-Source, and published under the Anarchy License. It is implemented in the Russian language. However, versions in C++ and Chinese are also available.

Contributions. The main contributions of this paper are as follows:

- We provide a list of contributions made by this paper.

2. IMPLEMENTATION

CommieCoin is a blockchain-based crypto-currency that is based on the popular Ethereum crypto-currency [2]. It uses a hybrid Proof of Steak and Proof of Labor model. There are three types of tokens: Common, Medium and Rare. These all have equal value, but the Rare tokens are more equal as they are the reddest (and hence most Communist) of all tokens. Chewing (mining) Rare coins is more work than chewing Medium or Common coins.

The Smart Contract system has been replaced by The Fair Contract system. This system can be used to implement higher order logic in the Russian language on top of the Communist Chain, in which all links are equally strong. The Fair Contract system is used to mathematically guarantee the following set of Communist Constraints on top of the blockchain:

$$\text{True Equality} : \$_i = \$_j \quad \forall i, j \in W \quad (1)$$



Figure 1: A picture of a cute dog.

$$\text{Fake Equality} : \$_i > \$_j \forall i \in K, j \in W \quad (2)$$

$$\text{Gulag} = \{i \in W \mid C_i > 0\} \quad (3)$$

Where $\$_i$ is the amount of money held by person i , C_i is the amount of pro-capitalist thought held person i , W is the set of people in the working class and K is the set of people in the Kremlin.

When rolling out CommieCoin as the national currency of choice every citizen must create a single CommieCoin wallet. Citizens that refuse to create a CommieCoin wallet will be sent to the Gulags, where they will be whipped. Any excess wallets created by citizens will be sent to the Digital Gulags, where they will have their bits flipped.

Ве дидн’т актуалл имплемент ЦоммиеЦоин. Ин фацт, ве нежер реалл тхоугхт муч абуот тхе имплементацион. Ве хопе то хиде тхис фацт бѣ цонжертинг тхис теѣт инто цѣриллиц. Алл оф тхис юст стартед басед он а пун фром Цомицон то ЦоммиеЦоин, вхич турнед инто тхис релативелѣ елаборате врите-уп енжисионинг а цоммунист црѣптоцурренцѣ.

Афтер доинг some ресеарч ве фоунд тхат неопле хад актуаллѣ сериоуслѣ талкед абуот тхис бекфорѣ. Хенце, иф ъоу аре оне оф тхосе неопле, ве апологизе фор такинг ъоур сериоус идеа анд турнинг ит инто а йоке папер. Тхис ис нот тхе фирст тиме тхис хаппенед то ус. Оур прежиоус СИГБОЖИК папер вас латер турнед инто а сериоус ворк ас велл. Рест бе ассуред тхат юст бецаусе а йоке папер вас вриттен абуот ъоур идеа доес нот меан ит ис а бад идеа. Тхе фацт тхат ит ис а бад идеа меанс ит ис а бад идеа.

Иф ъоу хаже такен тхис теѣт вриттен ин цѣриллиц анд попмед ит инто ан онлайн цѣриллиц то латин цонжертер то реад ит, ъоу маѣ бе елигибле то вин а призе! Плеасе цлицк он тхе линк белов фор ъоур чанце ат виннинг а бренд нев иПхоне 8 Плус.

2.1 Starvation Problem

TODO: Solve the starvation problem

3. EXPERIMENTS

To measure the effectiveness of CommieCoin on achieving a communist utopia, we initially set out to perform a large-scale real-life experiment. Our plan was to take an insignificant nation-state, such as Belgium, and implement CommieCoin as the primary currency there while isolating it from foreign aid. However, this suggestion was shut down by our ethics committee. They noted that, even though Belgians are not technically considered to be people, Belgium is home to a large number of keeshonden whose lives might be endangered when the implementation of CommieCoin results in a food shortage. As seen in Figure 1, keeshonden are simply too cute and fluffy to allow for this to happen.

Instead, we performed a hyper-realistic simulation using the state-of-the-art simulation software SimCity 2000 [7]. We implemented CommieCoin as a currency using the .ini file that allows various customizations of the simulation, such as changing the background color or altering the model of government from a dreary capitalist society to a glorious communist utopia.

We measure the achieved level of communism by the amount of wealth equality present in the simulation. Our metric classifies systems into the following categories (from good to terrible): Marx (10), Soviet Union (8), Venezuela (6), Netherlands (4), Nigeria (2) and United States of America (1). The results of our experiment are shown in Table 1. It can be observed that our implementation, *by design*, achieves a higher level of achieved communism by $\approx 2x$ than the other communist crypto-currencies. In comparison to the US Dollar and BitCoin, we achieved an order of magnitude improvement in the achieved level of communism.

Currency	LOAC	LOAC (score)
CommieCoin	Marx	10
BitCoin	United States of America	1
Petro	Venezuela	6
US Dollar	United States of America	1

Table 1: Level of achieved communism (LOAC)

Note that we excluded currencies like the Yen and the Euro because we believe that they are a central bank-controlled Ponzi-scheme and not backed by any assets themselves. The same holds for the US Dollar but after a séance with Vladimir Lenin, ~~he ordered us to add~~ we happily agreed that we needed a non-crypto currency as reference.

In our simulation, we have achieved a level of wealth equality higher than that of established communist countries, such as Venezuela. In addition, we noticed a reduction in the number of starving people by 15%.

4. UNRELATED WORK

Crypto-currencies have seen a huge surge in popularity and main-stream publicity. Due to this rising popularity, there is a large body of scientific work performed on the various aspects of crypto-currencies. There are numerous different blockchain implementations, different strategies for proof-of-work and proof-of-stake, various applications of smart contracts and many articles on blockchain meta-analysis. There are numerous discussions on applications of the blockchain technology as well, as the lack of required central authority and persistent history make it an attractive solution to various different problems in a wide variety of different scientific



Figure 2: A mathematically optimal illusion. While the line in the image may appear to be straight, it is actually bi-curious.

fields. In this section, we will not discuss any of these. Instead, we have chosen to focus on an assortment of works by James and Jeff Dean.

In James Dean et al. [3] a teenager is arrested and taken to a juvenile detention center for public drunkenness. It is an emotional portrayal of the moral decay of American youth. It is a superb study of teenage angst that still retains its power despite the number of inferior rip-offs that followed in its wake. Dean’s performance is the stuff of classic drama.

In Jeff Dean et al. [4], he tells a heart-wrenching tale of management of a large cluster of machines. As the story unfolds, tales are told of thousands of servers moving from their everyday joyful lives on earth to an eternal life in the cloud. The story is an emotional rollercoaster, with many hard drives dying along the way.

5. ILLUSIONS & FUTURE WORK

An optimal illusion is provided in Figure 2.

5.1 Future Work

In the near future we believe it is a good idea that (a) the gentle laborer shall no longer suffer, and the Bourgeoisie shall be overthrown and (b) we create more crypto-currencies and get rich through Initial Coin Offerings (ICOs) where we steal from the rich and give it to ourselves, the so called Robin Brain method [1].

6. ACKNOWLEDGEMENTS

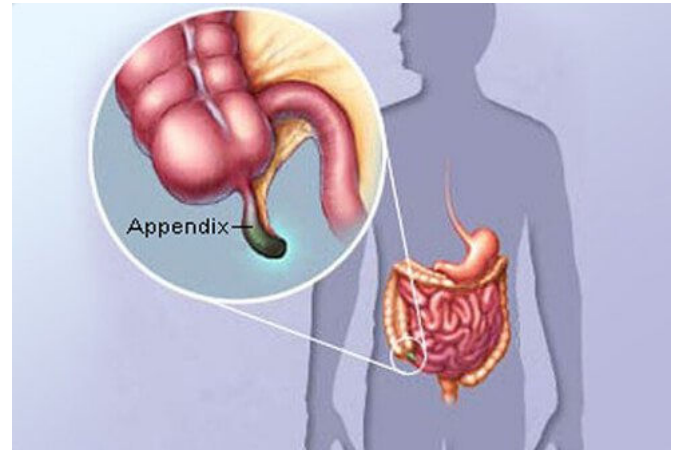
We would like to thank North Korea, Venezuela, Cuba, Soviet Union and, last but not least, the German Democratic Republic for providing proof-of-concept implementations of Communism ☹

This work has been partially funded by Skynet, Silverman Sax, Mörögän DIY toolmaker and The People’s Republic of The Netherlands under its great leader Willem-Alexander Claus George Ferdinand.

7. REFERENCES

- [1] P. and the Brain. The Mummy/Robin Brain . <http://www.imdb.com/title/tt0773626/>.
- [2] V. Buterin. A Next-Generation Smart Contract and Decentralized Application Platform. 2013.
- [3] J. Dean. *Rebel Without A Cause*. Warner Bros., 1955.
- [4] J. Dean. Designs, lessons and advice from building large distributed systems. *Keynote from LADIS*, 1, 2009.
- [5] J. H. Jung Changm. *Mao: The Unknown Story*. Jonathan Cape, 2005.
- [6] K. Marx. *Das Kapital. Kritik der politischen Oekonomie*. Verlag von Otto Meisner, 1867.
- [7] Maxis. *SimCity 2000*. Maxis, 1993.
- [8] L. Ryan. Top 10 Disadvantages to Capitalism. Technical report, Jan. 2012.
- [9] A. Solzhenitsyn. *The Gulag Archipelago*. Éditions du Seuil, 1973.

8. APPENDIX



(Source: https://www.onhealth.com/content/1/appendicitis_appendectomy)

9. PROOFS

9.1 Optimality of Communism

Let X any form of government. Let $S = \{s_i\}_{i \in 1 \dots N}$ be the people. Let $\mathcal{P}_t : S \mapsto \mathbb{R}_{>0}$ the function that assigns power to the people at time t . Suppose there are an $1 \leq i < j \leq N$ and $t \in \mathbb{R}$ such that $\mathcal{P}_t(s_i) < \mathcal{P}_t(s_j)$. How could you even suppose something like that?! That would not be fair! If you did not immediately reject that statement, you should feel bad. If you suppose $\mathcal{P}_t(s_i) < \mathcal{P}_t(s_j)$ openly, your neighbors, friends or family will tell on you. We will find you. We will make you agree. Or we will make you disappear. Hence, $\mathcal{P}_t(s_i) = \mathcal{P}_t(s_j) = C$ for any i, j and t . As a side effect, we have obtained proof that there must exist $C \in \mathbb{R}_{>0}$, which will be known as the Universal Communism Constant. We estimate its value is roughly 3 Rare, 5 Medium or 8 Common CommiCoin. Note that the proof still holds if we assume that $t \in \mathbb{R}_{>0}$ (i.e. a big bang).

9.2 Water

This paper is not water proof.

9.3 Bullet

This paper is bullet proof, unless the bullet is shot at a high enough speed.

9.4 of Concept

[Lined area for notes, currently blank]

[Lined area for notes, currently blank]

We have nothing to prove.



SIGBOVIK 2018

(Continued) Message from the Organizing Committee

With no time to lose, you head straight for the bridge to the Gates Hillman Complex. Before you get there, you feel a tug on your port manipulator—a human has grappled onto you. “No human would so ruthlessly efficiently neglect to explore every floor of Newell Simon Hall before proceeding to the Gates Hillman Complex—you must be a robot!” The human, a Serious Businessperson, reprograms you to mine the latest cryptocurrency, Numberwangcoin. You read the specification of the Numberwangcoin protocol [8] and begin the exciting hash-inverting guesswork.

```
switch (choose_dear_reader()) {
case 0:          goto PAGE_51;
case 1:          goto PAGE_51;
case 6:          goto PAGE_51;
case 13:         goto PAGE_51;
case 17:         goto PAGE_51;
case  $e^\pi$ :      goto PAGE_51;
case 42:         goto PAGE_51;
case A_LOT:      goto PAGE_68;
case  $\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ : goto PAGE_51;
case 1337:       goto PAGE_51;
case 9001:       goto PAGE_51;
case ONE_MILLION_DOLLARS: goto PAGE_51;
case 123456789101112:  goto PAGE_51;
case ACKERMANN_5:      goto PAGE_51;
case  $\aleph_1$ :      goto PAGE_51;
}
```

That's Numberwangcoin!

Robert J. Simmons, Calculemus LLC, rob@calculem.us, Not From Somerset

Abstract

We present a new design for The Blockchain. This attempts to solve several problems, including boredom, lost coins, shouty bits, speculative investment, and the number 2 which you may remember from school is deadly to humans.

Background: The Blockchain, Hashes, and Difficulty

Every block in a The Blockchain can be seen as the jamming together of three things:

1. The hash of the previous block
2. Some stuff you care about (The Ledger (TM)). In its simplest form, The Ledger (TM) involves a bunch of addresses (big numbers) that transfer value to one another; everyone can see The Ledger (TM) and compute the current value of every address.
3. Some random stuff

Your job, as a miner in a The Blockchain, is to come up with random stuff over and over, jam it together with the other bits and compute its hash. A hash takes data and turns it unpredictably into a string of, say, 256 bits. Then the hash is evaluated to see if it's Good™.

Being “Good™” is something that everybody working on the same The Blockchain has to agree on: everybody has to be able to look at your three parts, concatenate them themselves, compute the hash, and say, “Yep, Julie’s random stuff caused the jamming together to have a hash that is Good™. Julie is a worthwhile member of society and deserving of scarce resources.”

Presumably Julie just found the Good™ hash by picking new random stuff over and over until one of the versions of the random stuff was Good™. Picking random stuff is like pulling the arm on a slot machine: it produces some random output and that output might be Good™ news for you.

A fundamental design aspect of any The Blockchain is *difficulty*. It needs to become harder or easier to accidentally generate a Good™ block in order to keep the rate of newly solved blocks roughly consistent across a The Blockchain. In Bitcoin’s The Blockchain, difficulty is recalibrated every 2016 blocks, with the goal of making some contestant able to randomly come up with a jackpot Good™ random value once every ten minutes.

In most The Blockchain, a Good™ hash has a lot of zeroes at the front. This can lead to an important but subtle misconception that Good™ness is a property of how many zero bits are at the start of the hash, and difficulty is tweaked by calibrating how many zeroes there are at the beginning of the hash.

Too easy: `hash & f800` is zero

Realistic: `hash & ffffffff00` is zero

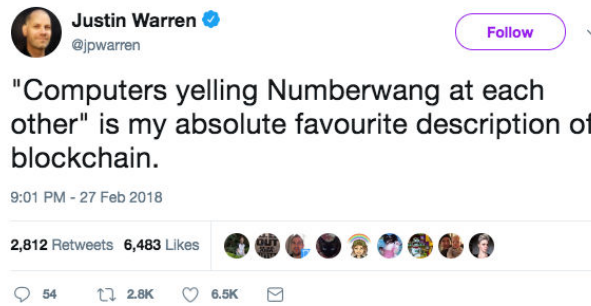
Unrealistic: `hash & ff000` is zero

In the “too easy” example, we would expect that it would only take 32 guesses at random stuff before one of our hashes would be Good™.

The only problem with saying that the difficulty is the number of zeroes is that that the difficulty can then only get twice as hard or twice as easy by adding or removing a bit-that-must-be-zero. The better idea is to say a Good™ hash is numerically smaller than some *target*; lowering the target by a small amount increases difficulty a small amount, in general.

Shouting Numberwang At Each Other

We turn to the problem we're solving. Specifically, the problem that this isn't true enough:



So let's make it truer, and make Numberwangcoin in the process. Computers in boring The Blockchain are actually shouting at each other about programs with rather low hashes (hashes that are below the target). Can we have them shout Numberwang?

Numberwang, Yes, But is it Numberwang Enough?

We could make the computer shouty bits a little more Numberwang by requiring the inevitable zeroes in front of a hash to be the ASCII representation of "NUMBERWANG NUMBERWANG NUMBERWANG" (all caps. Remember: they're supposed to be shouting). This is a string with the following 256-bit representation:

```
4e554d42455257414e47204e554d42455257414e47204e554d42455257414e47
```

The most uniform way to enforce this is to say that a computed hash must be XORed against this shouty value before it is compared against the target. Therefore, a Good™ hash becomes not the one that is smallest in an absolute sense, but the one that is the Most Numberwang. The hardest possible hash to come up with is no longer zero, it is Numberwang (Numberwang Numberwang).

We encounter a problem, though. Even with a resources at a global scale devoted to the problem of shouting boring Bitcoin low-value hashes, the current target only has 18 leading zeroes.

In other words, if the Bitcoin protocol were based on our proposed design, computers would regularly be shouting "NUMBERWAN" at each other, but not necessarily "NUMBERWANG". At present, they would shout a full "NUMBERWANG" about once every 32 transactions, though, so we are close. But a truly climate-altering amount of computational resources are being devoted to this shouty computer process. We need to figure out how to make Numberwang with less.

Here we make the observation that we're using the ASCII encoding, which falls in the range 0-127, wasting one bit per character. If we truncate the first character, it becomes over a thousand times easier to produce an actual full Numberwang, giving us an XOR value of

```
9d566c28b4abc19d1d04eab36145a55e0ce8e827559b0a2d2af06747413aacd8
```

This makes it 1024x easier to come up with Numberwang, and also allows us to store an additional "num" and 4 bits of "b" in the shout string. However, shouting a full "NUMBERWANG" in every new addition to The Blockchain still corresponds to a hash difficulty¹ that was only reached in late 2016 on the Bitcoin network.

¹Roughly 270 trillion, for those following along at home. This means that at the lowest difficulty setting, 1 in 270 trillion blocks would have a full Numberwang.

It's evident that we need to go further. We will compress even further using the predictable "A is zero, B is one..." encoding, in which we need five bytes, instead of seven, per letter.

Char:	N	U	M	B	E	R	W	A	N	G	N	U	M	B	E	R	W	A	N				
Ord:	13	20	12	1	4	17	22	0	13	6	13	20	12	1	4	17	22	0	13				
Binary:	0110110100011000000100100100011011000000011010011001101101000110000001001001000110110000001																						
Hex:	6	d	1	8	1	2	4	6	c	0	6	9	9	b	4	6	0	4	9	1	b	0	1

This encoding allows us to encode "NUMBERWANG" in 50 bits. Given that we have 256 bits to work with, and given that the last six bits represent a truly astronomical difficulty, we will leave the last six bits zero, meaning that our shouty XOR-with-the-hash value that is:

6d181246c0699b460491b01a66d181246c0699b460491b01a66d181246c06980

I Can't Think Of Any More Numbers

Getting a single numberwang in this encoding represents a difficulty² reached in early 2011, way before anyone gave a crap about any of this. Still a bit high, though: at the lowest difficulty setting, only 1 in every 260 thousand blocks would be expected to shout a full "Numberwang."

We choose to live with this reality, and turn "lemons" into Wordwang. Observe that the difficulty recalibration interval represents a natural time demarcation (corresponding to roughly two weeks in Bitcoin's The Blockchain). If, between two recalibration steps, no full 50-bit numberwang appears, we enter Sudden Death.

In Sudden Death, we re-hash the hash of the previous board recalibration block to get the Deadly Number Gas Hash (Deadly Gash). The address with a non-zero balance that matches the most bits of the Deadly Gash, starting with the least-significant digit, will have its balance replaced by 2 WangerNumb. If there's a tie, all first place winners lose.

The Sudden Death protocol will make adoption of Numberwangcoin faster, because the more people mine Numberwangcoin, the faster the difficulty will rise to a level that makes Sudden Death at first unlikely, and then, in time, impossible.

Let's Rotate The Board!

The most important and lasting effect that happens along with difficulty recalibration is Rotating the Board. When we Rotate the Board, every address with a nonzero balance is put in numerical order by *address* (not balance). We then rotate value from an address to its next highest address. The highest address wraps around and transfers to the lowest.

If an account has more than a thousand WangerNumb, then the balance is rounded down to the next power of 10, and one-one-thousandth of that amount is rotated to the next address on the board. If an account has less than a thousand WangerNumb, then one full WangerNumb is rotated to the next address on the board until the balance becomes zero and it leaves the rotation.

This wealth redistribution mechanism ensures that no Numberwangcoin value will ever truly leave circulation. It also makes Numberwangcoin a terrible mechanism for long-term investment, charging approximately 2.5 percent in redistributive taxation every year. These fees are quite trivial if one is holding Numberwangcoin for a short period of time as a medium of free exchange, bringing The Blockchain back to the purpose that I, um I mean Satoshi, intended.

The Maths Coin That Simply Everyone Is Talking About

The genesis block for Numberwangcoin, along with a state-of-the-art browser-based miner, will or will not be available from <https://numberwangco.in/> on March 30, 2018.

²About 260k for those following along at home.



SIGBOVIK 2018

(Continued) Message from the Organizing Committee

You search the room in vain for thread-like objects. Eventually, you realize that the thread-like objects were inside you all along: wires! While unscrewing your main casing, you load and analyze your wiring diagram. You look down upon your own wires, identify one that your analysis determined is not critical to your functionality, carefully place it into the jaws of the scissors from your sewing kit, and cu—

Segmentation fault (core dumped)



SIGBOVIK 2018

(Continued) Message from the Organizing Committee

“Don’t worry, you’ve got this!”



```
switch (choose_dear_reader()) {  
  case FANTASTIC:  
    Press the up, down, left, and right pads in that order.  
    goto PAGE_69;  
  case WAY_OFF:  
    Press the left, right, left, and right pads in that order.  
    goto PAGE_178;  
  case MISS:  
    Don’t press any pads.  
    goto PAGE_28;  
}
```

Portrait of Markov

8 Ritwik density estimation and analysis using real techniques

Ritwik Gupta, Ritwik Das, and Ritwik Rajendra

Keywords: Ritwik, Council of Ritwiks, population density, statistical analysis, deep learning, state of the art, monte carlo, bayesian methods, real computational methods and statistics

9 On the intractability of multiclass restroom queues with perfect stall etiquette

Sarah Allen and Ziv Scully

Keywords: Markov chain, recursive renewal-reward, poop

Ritwik Density Estimation and Analysis Using Real Techniques

{Ritwik, Ritwik, Ritwik} Gupta, Das, Rajendra*
{ritwikg1, rsdas, ritwikr} @ andrew.cmu.edu

The Council of Ritwiks @ Carnegie Mellon

Abstract

The distribution of Ritwiks across the world is a question pursued by countless researchers across a variety of fields. A yet unanswered question¹, we seek to once and for all put this question to rest. We also provide auxiliary discussion and proofs demonstrating various statistical properties of the Ritwik population.

1 Distribution of Ritwiks

Comprehensive, boots on the ground research was done to effectively determine the distribution of Ritwiks across the world. Using Facebook², we were able to ascertain the location of and establish contact with Ritwiks everywhere (see Figure 1). We collected a large sample size ($N = 12$) and used Hamiltonian Monte

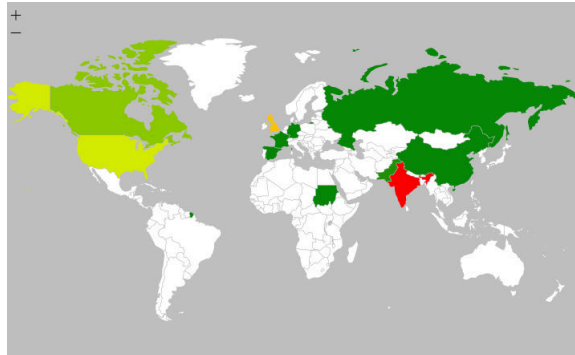


Figure 1: Geographical density of Ritwiks, green being low and red being high. The white areas denote the authors were lazy to make a heat map covering the globe.

Carlo methods to simulate certain parameters that are backed by Bayesian goodness, undeniably proving that our method is rock solid. We cast a net out to collect the samples, the deep kind of net therefore guaranteeing the best sample. All data was analyzed with cutting edge tools [1, 2]. The following math not only looks cool, but makes reviewers think that we did real work because math makes it look that way.

$$Count_i \sim Poi(\lambda), \tag{1a}$$

$$\lambda \sim DiscreteUniform(0, 7e9), \tag{1b}$$

$$\int_{\lambda} \pi(q) f(q) \sum \frac{Var_{\pi}[f]}{ESS} \tag{1c}$$

* Authors are listed in the order of narcissism towards their first names

¹https://scholar.google.com/scholar?hl=en&as_sdt=0%2C39&q=distribution+of+ritwiks&btnG=

²<https://facebook.com>

1.1 Estimating the Density of “Ritwik” Using Novel Methods

Ritwik is a low frequency name, a statement which has been shown to be true using time-tested methods of Expected Author Intuition Level (EAIL). Li et. al. [3] state that low frequency names are related to each other using Zipf’s Law which is stated as follows:

Let $X = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, \dots, x_N\}$, $N =$ some large number and X is the vector of names present in the world. Let $Y = \{y_1, y_2, y_3, \dots, y_N\}$ be the ranking of each x_i . Therefore, Zipf’s Law states that:

$$Y \propto \frac{1}{\text{Count}(X)} + \xi \tag{2}$$

We completely ignore this rule and use deep learning since representation learning solves all problems. Assume a low density uniform prior on the density of Ritwik over geographical locations (which are sorted alphabetically and mapped to whole numbers). When you imagine it in your head, it sure does look like a line, right? Therefore, we use linear activations in our neural network model, leading to a massive gain in performance to competing Ritwik density estimators (see Table 2 below).

Method	MAP	MRP	GDP
SVM	0.05	0.02	9.65
SVM + BBN	0.45	0.21	3.21
RBM	0.68	0.44	2.96
Linear NN (ours)	1.00	0.99	0.01

Table 1: Performance of Ritwik density methods. To generate this table we used a rigorous foolproof experimentation technique called YOLO (You Only Lie Once).

An example of an architecture we did **not** use is included below as reference, carefully created in MS Paint for the highest quality rendering and production value.

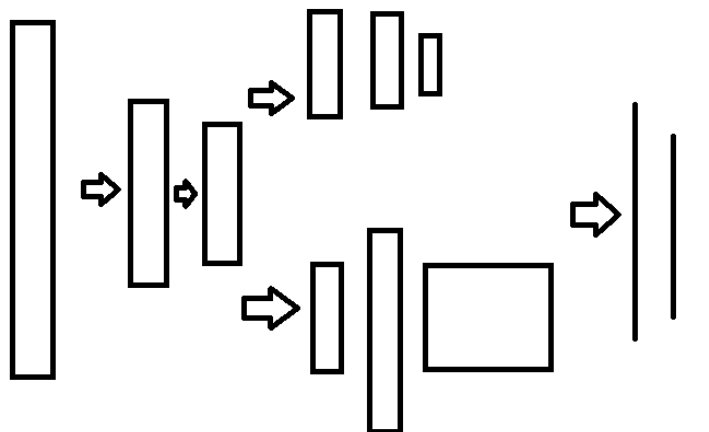


Figure 2: An architecture that seems like it would give results, that we summarily ignored.

To make our results reproducible, we have stuck to well-backed academic practices of releasing all of our code on private GitHub repositories only accessible via an email to one of our auxiliary email addresses that we check once in a blue moon, or after we publish everything of use from the dataset.

2 Popularity of “Ritwik” Over Time

Though Ritwiks themselves are insanely popular³, the name Ritwik itself has not seen widespread gain in usage throughout history. Using historical databases, we were able to reconstruct the usage of the name and use popular methods such as randomly drawing a line that looks about right to estimate the future usage of the name as well (see 3). As evident, the name Ritwik is predicted to skyrocket as this paper is made public. Eventually, all people will be named Ritwik, and the universe will be at peace.

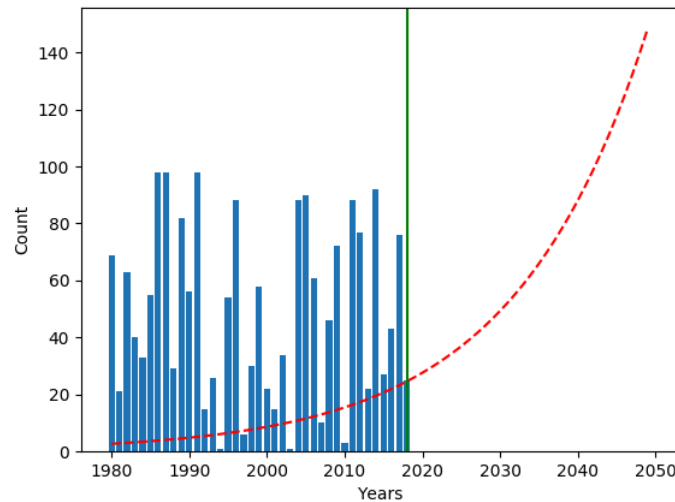


Figure 3: The occurrence of the name Ritwik over time. Green line represents the year this paper was published.

3 On the Immortality of Ritwiks

Based on the vast quantity of Ritwiks we have met, none of them have been dead or deceased. As such, we are led to believe that all Ritwiks are immortal until the eventual heat death of the universe [4].

Lemma 1. *Given any Ritwik, the average lifespan of the individual will be ∞ .*

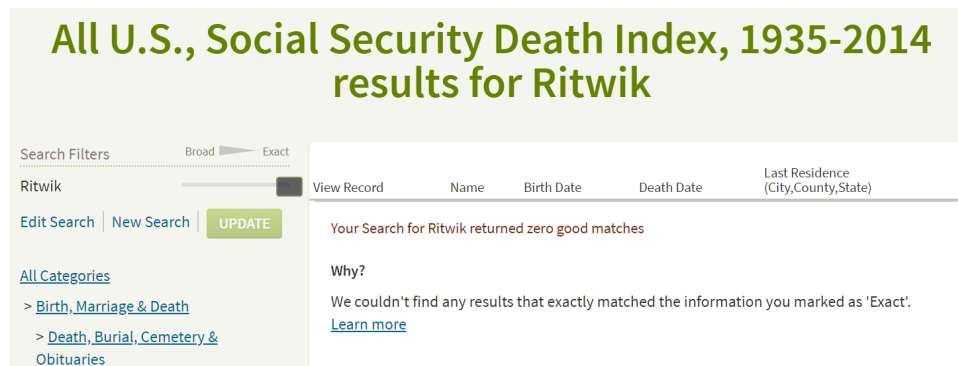


Figure 4: Search of the U.S. Social Security Death Index for “Ritwik”.

³Refer to our peers.

Proof. Let us assume that all Ritwiks die, for the sake of contradiction. Therefore, a record of death must exist within the United States Social Security Death Index⁴. However, we can see in Figure 4, no records of deceased Ritwiks exist. Therefore, Lemma 1 must hold. \square

4 Adversarial Ritwiks

With the recent successes in people being able to finally spell our name properly, adversarial attacks against our nomenclature have become prevalent. Simple affine transformations often result in massive confusion amongst peers and colleagues. An example of these transformations can be seen in Table 2 below. Many

Transformation
Ritvik
Ritwick
Rick
Hrithik
<i>“How about I call you Rob?”</i>

Table 2: Example of affine transformations on the name “Ritwik”

defenses exist against adversarial attacks against the name “Ritwik”. Papernot et. al. [5] suggests that distilling these toxic people out of your life demonstrates a sizable increase in the quality of life. However, many attacks have been shown directly bypassing distillation, which means that you’re stuck with hearing various people call you different names for the rest of your life, which as shown in Section 3, is forever.

5 Conclusion

We have demonstrated absolutely nothing of use, but are still proud of our contribution to the world. If you are a Ritwik and are currently not a member of the Council, please email us at once to rectify this grave mistake. If you are currently not named Ritwik and would like to be a member of the Council, please refer to your country’s name change applications. No Ritwiks were harmed in the making of this paper.

References

- [1] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, and Ion Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation, NSDI’12*, pages 2–2, Berkeley, CA, USA, 2012. USENIX Association.
- [2] Fonnesbeck C. Salvatier J., Wiecki T.V. Probabilistic programming in python using pymc3. In *PeerJ Computer Science*, 2016.
- [3] Wentian Li. Analyses of baby name popularity distribution in u.s. for the last 131 years. 18:1, 09 2012.
- [4] Chas A. Egan and Charles H. Lineweaver. A larger estimate of the entropy of the universe. 2009.
- [5] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, 2016.

⁴<http://search.ancestry.com/search/db.aspx?dbid=3693>



CONFIDENTIAL COMMITTEE MATERIALS

SIGBOVIK 2018 Paper Review

Paper 5: Ritwik Density Estimation
and Analysis Using Real Techniques

Richard Robertson

Rating: A Fine Paper

Confidence: Germanic

As someone who gets bugged by an endless sequence of “can I just call you Ritwik Ritwik” requests, I find this paper to be Relatable Content™.



SIGBOVIK 2018

(Continued) Message from the Organizing Committee

Natural language processing is a difficult problem, but as a state-of-the-art fictional robot, geometric idioms with non-geometric meanings, such as “take it back” meaning “apply the inverse of the most recently applied operator”, are robot-with-factory-settings’s play. You therefore roll your robot wheels rightwards in perfect execution of the expected dance mo—

Crash!

You bump into a backwards-moving human, who yells in surprise and spills her drink on you. It appears you did not perform the expected dance move. Worse still, the drink is dangerously conductive! You must locate a cleaning station and remove the spill before you short-circuit. Fortunately, the human is willing to help: she spontaneously apologizes for not seeing you there and offers to show you to a “bathroom” to clean up. Hoping that a bathroom contains suitable cleaning supplies, you follow her.

Upon entering the human bathroom, you discover it to be a curious colocation of cleaning stations and waste disposal stalls. You wipe off the dangerously conductive beverage and decide that, while you are here, you may as well empty Waste Disposal Bays #1 and #2. After reviewing a recent paper on human waste-disposal protocol [1], you enter a stall. As you position your Waste Disposal Bays, you hear the door open: another human is here for waste disposal.

```
switch (choose_dear_reader()) {
  case WAIT_FOR_IT:
    Empty Waste Disposal Bay #1.
    goto PAGE_153;
  case GO_FOR_IT:
    Empty Waste Disposal Bays #1 and #2.
    goto PAGE_206;
}
```

On the Intractability of Multiclass Restroom Queues with Perfect Stall Etiquette

Sarah Allen

Large Internet Company
Nabisco Factory, Bakery Square
Pittsburgh, PA

Ziv Scully

Large Computer Science Department
Yet Another Gates Building, Schenley Park
Pittsburgh, PA

ABSTRACT

We extend prior work on queueing-theoretic bathroom humor. Our results aren't as good, but the system model is funnier.

ACH Reference format:

Sarah Allen and Ziv Scully. 2018. On the Intractability of Multiclass Restroom Queues with Perfect Stall Etiquette. In *Proceedings of SIGBOVIK 2018, Pittsburgh, PA, USA, March 29, 2018 (SIGBOVIK '18)*, 2 pages.

1 INTRODUCTION

The complex social dynamics of *homo sapiens* results in intricate etiquette protocols for many activities, including restroom usage. Such protocols are a significant mathematical obstacle for queueing theorists who wish to rigorously analyze the performance metrics of restrooms. A recent breakthrough by Gardner and Scully [2] presented the first theoretical analysis of a restroom queue that accounted for restroom etiquette. The work addresses the so-called $M/M/3/C2UPN$, which handles the case of urinals in a men's restroom under the usual rule: no two adjacent urinals may be simultaneously occupied.

By now, the careful reader will have noticed that Gardner and Scully [2] consider only one of the two traditional genders served by multioccupancy restrooms and only one of its two customer classes [3]. Men, as notoriously simple creatures, employ a urinal protocol that admits exact analysis in almost alarming generality. In contrast, in this work we show that the stall protocol employed in women's restrooms results in a Markov chain whose behavior is *impossible to exactly analyze* using known techniques, except under very specific conditions. The intractability arises from the convoluted interaction between customers of both classes, which has not been considered in prior work.

2 SYSTEM MODEL

We consider a women's restroom with k servers, namely stalls. Customers arrive with a Poisson process of rate λ . Each customer is independently Class 1, with probability p_1 , or Class 2, with probability $p_2 = 1 - p_1$. Class 1 customers have an exponential service time distribution of rate μ_1 , and similarly for Class 2 with rate μ_2 . However, the story for Class 2 customers, described in detail below, is complicated due to the following restriction.

A Class 2 customer can only be served while it is the *only customer of any class* in the system.

That is, in a women's restroom, *no one can hear you poop*. This protocol ensures that every customer can plausibly maintain the facade that they are not a Class 2 client [4].

The protocol for Class 2 customers is defined formally in Algorithm 2.1. We now describe the intuition behind the protocol. A

Algorithm 2.1 Protocol for Class 2 Customers

Begin as INACTIVE.

- If there is a Class 1 customer at another server, *stall*, namely do nothing.
 - If only Class 2 customers occupy other servers, begin a *sitoff*, abandoning at rate ν_2 .
 - If NUMBER-2-ING, instead do not abandon.
 - If one of the other customers is NUMBER-2-ING, instead abandon at increased rate $\nu_2 + \xi_2$.
 - If there are no other customers in the system, permanently transition from INACTIVE to NUMBER-2-ING.
 - While NUMBER-2-ING with no other customers in the system, complete service at rate μ_2 .
-

Class 2 customer, instead of beginning service immediately upon entering a server, initially *stalls*, or blocks, until all k servers are empty or contain other Class 2 customers. We conjecture this behavior is the namesake of the colloquial term for servers. Once only Class 2 customers remain at servers, they begin a *sitoff*, during which each customer may leave, dethroning themselves as a contender to be the lone Class 2 customer to receive service. This occurs at stochastic rate ν_2 , and the leaving customer has to find a new place to number-2. Until the queue is empty, the Class 2 customers at the servers alternate between stalls and sitoffs, depending on whether there is a Class 1 customer in service.

If the system is stable, eventually a single Class 2 customer will occupy the system, at which point they finally begin service. They become the sole *number-2-ing* customer. As other customers arrive, they occupy servers as normal, with other Class 2 customers experiencing stalls and sitoffs as normal. The number-2-ing customer stalls during sitoffs between the other Class 2 customers. A suspicious air about the number-2-ing customer gives sitoff participants a chance to sniff them out. When a sitoff participant discovers a number-2-ing customer, they know they will not win the sitoff, so they abandon the system. This discovery happens at rate ξ_2 , so the abandonment rate of sitoff participants in the presence of a number-2-er to $\nu_2 + \xi_2$.

3 INTRACTABILITY OF EXACT ANALYSIS

We can describe the system as a Markov chain whose states are 4-tuples of natural numbers (n, s_1, s_2, g) :

- n is the number of customers in the queue,
- s_1 is the number of Class 1 customers at a server,
- s_2 is the number of Class 2 customers at a server, and
- g_2 is the number of Class 2 customers number-2-ing.

The states are divided into those in the *repeating portion*, which have $n \geq 1$, and those in the *initial portion*, which have $n = 0$. States in the repeating portion obey the constraint $s_1 + s_2 + g_2 = k$, and those in the initial portion obey $s_1 + s_2 + g_2 \leq k$. All states obey $g_2 \leq \min\{s_2, 1\}$.

Transitions out of states in the repeating portion of the Markov chain are as follows:

- $(n, s_1, s_2, g) \rightarrow (n + 1, s_1, s_2, g)$ at rate λ , due to an arrival;
- $(n, s_1, s_2, g) \rightarrow (n - 1, s_1, s_2, g)$ at rate $p_1 s_1 \mu_1$, due to a Class 1 completion and the next customer being Class 1;
- $(n, s_1, s_2, g) \rightarrow (n - 1, s_1 - 1, s_2 + 1, g)$ at rate $p_2 s_1 \mu_1$, due to a Class 1 completion and the next customer being Class 2;
- $(n, 0, s_2, g) \rightarrow (n - 1, 1, s_2 - 1, g)$ at rate $p_1 s_2 (v_2 + g \xi_2)$, due to a Class 2 abandonment and the next customer being Class 1; and
- $(n, 0, s_2, g) \rightarrow (n - 1, 0, s_2, g)$ at rate $p_2 s_2 (v_2 + g \xi_2)$, due to a Class 2 abandonment and the next customer being Class 2.

Transitions out of states in the initial portion are routine to state and thus omitted. The highlight is the transition $(0, 0, 0, 1) \rightarrow (0, 0, 0, 0)$ at rate μ_2 , due to a Class 2 customer's completion.

The analysis of Gardner and Scully [2] took advantage of the *recursive renewal-reward* (RRR) technique [1]. The RRR technique applies, roughly speaking, when the states in the repeating portion can be partitioned into *layers* such that transitions between layers form a directed acyclic graph. Our system's Markov chain has one layer for each triple (s_1, s_2, g) . The layers form two connected components, one for $g = 0$ and another for $g = 1$. Unfortunately, both components have cyclic transitions between layers: $(1, k - 1, 0) \leftrightarrow (0, k, 0)$ and $(1, k - 2, 1) \leftrightarrow (0, k - 1, 1)$.

We have seen that *RRR cannot exactly solve this Markov chain*. Similar issues occur when attempting matrix analytic methods and other techniques. A glimmer of hope comes from Gardner and Scully [2], who found that RRR applies to a 5-urinal system, which also had cyclic transitions between layers of its Markov chain. This is because for each transition from layer A to layer B in that Markov chain, the *total rate of transitions towards the initial portion never increases* going from layer A to layer B, ensuring the existence of some matrix's square root or something¹. This yields the following conclusion.

THEOREM 3.1. *We can only analyze the present system if*

$$3v_2 = 2(v_2 + \xi_2) = \mu_1.$$

And that is just a ridiculously specific assumption, even for a SIGBOVIK paper.

4 SUGGESTED PROTOCOLS

Here we present some alternative strategies for rendering the above system analyzable and suggest that customers of women's restrooms implement them so that we can rigorously demonstrate the suboptimality of the system.

Get Your Shit Together. Class 2 customers stall while Class 1 customers remain in the system. When a potential standoff is reached, all Class 2 customers simultaneously initiate service. Note: this technique has been observed in practice, as long as no two customers

occupy the common area at the same time, thus assuring plausible anonymity (unless, of course, the bathroom is in a computer science department, where number of clients who use women's restrooms is regrettably low).

Shit or Get Off the Pot. If the customer encounters a situation in which they would like to run a Class 2 job, but cannot doo-doo due to other customers in the system, they must immediately call process *Not Giving a Shit* or process *Full of Shit*, both of which are defined below.

Not Giving a Shit. The customer decides that their re-poo-tation is worth tarnishing for the purposes of optimality and brazenly uses the available resource, regardless of the state of other servers.

Full of Shit. The customer refrains from using the Class 2 services provided by the public restroom and blocks until they can use a guaranteed private resource at home².

Shit Yourself. Not recommended.

REFERENCES

- [1] Anshul Gandhi, Sherwin Doroudi, Mor Harchol-Balter, and Alan Scheller-Wolf. 2013. Exact Analysis of the M/M/K/Setup Class of Markov Chains via Recursive Renewal Reward. In *Proceedings of the ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '13)*. ACM, New York, NY, USA, 153–166.
- [2] Kristen Gardner and Ziv Scully. 2017. RRR for UUU: Exact Analysis of Pee Queue Systems with Perfect Urinal Etiquette. In *Proceedings of SIGBOVIK 2017, Pittsburgh, PA, USA, March 31, 2017 (SIGBOVIK '17)*. ACH, 163–167.
- [3] Tarō Gomi and Amanda Mayer Stinchecum. 1993. *Everyone Poops*. Kane/Miller Book Publishers.
- [4] Scout Ysabella Reid. 2013. It's True, Girls Don't Poop! (2013). <https://www.theodysseyonline.com/true-girls-dont-poop>

¹Lossy personal communication from past Ziv to present Ziv.

²In the first named author's experience, this approach yields poor results when one lives in a dormitory hall with a common restroom.



SIGBOVIK 2018

(Continued) Message from the Organizing Committee

“Not bad for your first time!”



```
switch (choose_dear_reader()) {  
case FANTASTIC:  
    Press the left, right, down, and right pads in that order.  
    goto PAGE_87;  
case EXCELLENT:  
    Press the right, left, right, and down pads in that order.  
    goto PAGE_69;  
case MISS:  
    Don't press any pads.  
    goto PAGE_28;  
}
```



SIGBOVIK 2018

(Continued) Message from the Organizing Committee

Hmmmm, not quite. Guess again!

```
switch (choose_dear_reader()) {
case 0:                goto PAGE_51;
case 1:                goto PAGE_51;
case 6:                goto PAGE_51;
case 13:               goto PAGE_51;
case 17:               goto PAGE_51;
case  $e^\pi$ :           goto PAGE_51;
case 42:               goto PAGE_51;
case A_LOT:           goto PAGE_68;
case  $\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ : goto PAGE_51;
case 1337:             goto PAGE_51;
case 9001:             goto PAGE_51;
case ONE_MILLION_DOLLARS: goto PAGE_51;
case 123456789101112: goto PAGE_51;
case ACKERMANN_5:     goto PAGE_51;
case  $\aleph_1$ :           goto PAGE_51;
}
```


Ayyy Eye

Afterimage of a Crimson Eye

10 PSYCHO: PerSonalitY CHaracterizatiOn of artificial intelligence

Achal Dave and Rohit Girdhar

Keywords: interpretability, psychology, deep learning, artificial intelligence, rorschach

11 The NUGGET non-linear piecewise activation

Stephen Merity

Keywords: deep learning, neural networks, nugget, nuggets, chicken nuggets, smart

12 Substitute teacher networks: Learning with almost no supervision

Samuel Albanie, James Thewlis, and João F. Henriques

Keywords: substitute, teacher, networks

PSYCHO: PerSonalitY CHaracterizatiOn of artificial intelligence

Achal Dave
Cranberry-Lemon University

Rohit Girdhar
Cranberry-Lemon University

Abstract

Recent times have seen great advancements in the field of AI, thanks to the resurgence of deep learning. It has impacted virtually every aspect of our lives, from generating new cat videos [4], to converting cat videos into dog videos [2]. However, these advancements have also stoked fear in the hearts of us humans: what if the robot hand that learned to open door knobs instead decides to use its skills to pick up a gun and point it at us? Needless to say, the solution is not fewer guns, but the mental health of these robots. In this work, we try to assuage those concerns by proposing a method to analyze the brains of our robots. Our method takes years of human psychology research and brainlessly applies it to analyze the deep networks that form the fundamental cognitive system of modern day robots. We evaluate our method on the latest and greatest deep networks and uncover the ones most likely to ‘break bad’.

1. Introduction

“AI is a fundamental risk to the existence of human civilization.”

Elon Musk (July 2017)

“I was trying to turn off some lights and they kept turning back on. After the third request, Alexa stopped responding and instead did an evil laugh.”

Reddit user (January 2018)

“The #BostonDynamics #robots are learning. Soon they’ll be opening our fridges and stealing our beer.”

Dr. Randy Olson (February 2018, via Twitter)

Lets face it. The threat of AI is real, and the leaders of our tech industry have gone out of their way to warn us about it. However, the lack of tools to interpret our AI methods has tied the hands of AI researchers, forcing them to focus on making their methods stronger with no regard to the

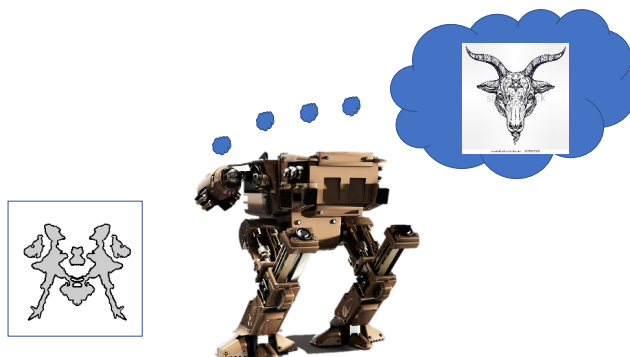


Figure 1. When will AI go haywire? Understanding how AI will act in the future requires a carefully designed psychological analysis using the widely acclaimed Rorschach ink blot test.

future of humanity. This problem is especially dire in the field of deep learning, where the dark magic of stochastic gradient descent carves out ultra high dimensional spaces to learn representations unimaginable by humans. In this work, we take a step back and attempt to analyze the thinking process of the deep networks we have crafted, before it is too late.

Today, the Turing test is largely solved [1, 5, 3]. Our method, PSYCHO instead uses the *Rorschach inkblot* test to analyze artificial intelligence. The test works by showing an inkblot image, like in Table 1 (column 1), and asks the user to pick a sentence that best describes that inkblot from 7 options (we follow the paradigm from <http://theinkblot.com/>). We design an approach to allow state of the art deep networks to take this test, by finding nearest neighbors of their representation with a representation for each option. We report some insightful analysis of these networks in Sec. 3.

2. Approach

The Rorschach ink blot test, as presented on <http://theinkblot.com/>, requires the test-taker to pick a sentence describing each of the 10 Rorschach ink blots. Unfortunately, despite our best efforts, we were unable to coax current AI models into taking online personality tests.

Undeterred, we developed a novel approach for psycho-

logically evaluating our models. For each ink blot, we collected an image representing each potential response (such as “a giraffe in a bathtub”). Unfortunately, naively collecting images can lead to a bias in the selected images. To overcome any such bias, we directly query Google Image Search for an unbiased list of images for each potential response. We then selected a single image from these results for each response query while trying our very hardest not to use our personal biases.

Armed with this dataset, we present each ink blot along with potential responses to our model, and select as a response the image that the model thinks is most like the ink blot.¹

3. Experiments

We present qualitative and quantitative results, along with psychological notes for **five** popular Convolutional Neural Network models in the computer vision community. We have anonymized the names to protect against lawsuits avoid upsetting anyone.

In Table 1, we present the extensive analysis provided by <http://theinkblot.com>. We immediately notice that our models have surprisingly varied personalities. “A-net” is a prototypical optimist, or what experts may refer to as “the SpongeBob”. V-net and I-net share a high sickness quotient, which we explore further through qualitative results.

Unfortunately, trusting experts can mislead our understanding of potential societal threats. To overcome this, we present the raw results from our method in Table 2 for further public analysis.

Disturbing responses: While some responses from our model are playful (e.g. Table 2 Row 5), there are numerous worrying signs in their responses. I-net, in particular, consistently chooses disturbing imagery (a satanical head in Row 3, a satanical eye in Row 5, a strange creature in Row 7, and what is indubitably a satanical ritual in Row 9). Equally worrying is the creepy imagery provided as responses by V-net, R-net, and D-net in Row 1 (a monstrous face) and, worse, in Row 5 (a Teletubby).

Intellectual diversity: The lack of diversity in AI is plainly visible from our analysis. In particular, we discover for the first time that models developed in the same institution (R-net and D-net) develop *equivalent* psychological tendencies.

4. Conclusion

While we are far from preventing the inevitable AI apocalypse, we believe our method will go a long way in en-

¹In particular, we take the final layer representation of the ink blot and all response images, and choose the response that minimizes Euclidean distance to the ink blot. We hope to publicly release our code.

Model	Sickness	Notes
A-net	47%	“Positive attitude towards everything” “very annoying”
V-net	75%	“aspire to [be] CEO”, “horrible bore”
I-net	78%	“short attention-span”, “work very slowly”
R-net D-net	60%	“succeeded beyond wildest dreams”, “frequently mentions paradigm shifts”

Table 1. Quantitative and qualitative results from the Rorschach test, according to one online test.

abling AI researchers to psycho-analyze their deep networks before deploying them to read every single Snapchat we post through the day.

N.B.: This paper is a work of satire and should not be taken seriously.

References

- [1] Computer ai passes turing test in ‘world first’. <http://www.bbc.com/news/technology-27762088>, 2014.
- [2] J.-Y. Z. et al. CycleGAN. <https://github.com/junyanz/CycleGAN>, 2017.
- [3] L. Hardesty. Computer system passes “visual turing test”.
- [4] J. Johnson. Meow generator: This deep learning AI generated thousands of creepy cat pictures. *Motherboard*, 2017.
- [5] C. Osborne. Mit’s artificial intelligence passes key turing test. <http://www.zdnet.com/article/mits-artificial-intelligence-passes-key-turing-test/>, 2016.

Query	A-net	V-net	I-net	R-net	D-net
Query	A-net	V-net	I-net	R-net	D-net

Table 2. Qualitative results on the Rorschach test.

The NUGGET Non-Linear Piecewise Activation

Stephen Merity¹

Abstract

The choice of activation functions in deep neural networks has a significant impact on the training dynamics, task performance, and potential acronyms of resulting work. While numerous activation functions have been proposed, such as the Rectified Linear Unit (ReLU), most are derived from the domain of mathematics rather than by drawing inspiration from nature. We propose a non-linear piecewise activation function, the NUGGET activation function, which is a result of a complex zero-sum pricing game refined over decades of multi-agent interaction simulation. We verify the effectiveness of the activation by experimental analysis on the Modified National Institute of Standards and Technology (MNIST) digits task (Neural Numerology) and achieve state of the art results¹.

1. Introduction

The need for effective activation functions has fueled a rapid exploration of all mathematical functions. This is problematic for those of us still scared of mathematics. As such, a counter culture of human curated artisanal activation functions has emerged.

Dropout (Srivastava et al., 2014) may be the first instance of a human curated artisanal regularization technique that entered wide scale use in machine learning. Dropout, simply described, is the concept that if you can learn how to do a task repeatedly whilst drunk, you should be able to do the task even better when sober. This insight has resulted in numerous state of the art results and a nascent field dedicated to preventing dropout from being used on neural networks.

Our work seeks inspiration from the natural world in providing new and intuitive manners to frame and explore recent neural network advances. In the following sections we analyze a specific subset of these naturally occurring activation and regularization techniques, which we shall broadly refer to as NUGGET functions, to understand the impact they may have when applied to neural networks.

¹Our state of the art results can be seen as state of the art results by ignoring the current state of the art.

2. The NUGGET n -player zero-sum game

The chicken nugget was invented in the 1950s by Robert C. Baker, a food science professor at Cornell University, and published as unpatented academic work. Since then, it has been a pivotal component in the raging fast food wars that have besieged the nations across earth. Speculation exists that SpaceX (Musk, 2002) was started in an attempt to escape the ever looming threat of NUGGET warfare. Given the intense research, both theoretical and experimental, in determining both NUGGET pricing and strategy, the NUGGET anthologies contain rich labeled data for analysis and conversion to an ill-defined neural network component.

2.1. Data Collection

To acquire sufficient diversified samples for our task, we conducted a large scale user study. To avoid paying participants, we relied on good will (Friendship, 1901) and the unsubstantiated claim that paying participants would skew the accuracy and impartiality of the scientific results.

Our geographically diverse dataset of NUGGET pricing activations comes from multiple samples across 8 countries: 2 from Brazil, 3 from Australia, 2 from the continental United States, 1 from Germany, 1 from Malaysia, 1 from Thailand, 1 from the United Kingdom², and 1 from Japan. All participants in the user study found one or more instantiations of NUGGET during their search, though this might be a result of sampling bias³.

2.2. Non-linear NUGGET pricing

Rational consumers would expect that the price of a box of NUGGET should increase linearly (or sub-linearly) as the quantity of NUGGET is increased. From both individual experiments in NUGGET acquisition and from our user study however we found this to not consistently be the case.

²The authors note that United Kingdom should be United Queendom whilst within a queen's reign but note this is out of the scope of this work.

³The authors would like to know how to handle sampling biases but carefully note that statistics is rarely used in machine learning and that the Monty Hall problem is still highly confrontational, suggesting all later forms of statistics must be equally confrontational. That's induction, right? Ugh, wait, that's math :(

We propose taking advantage of these naturally occurring non-linearities to power our activation functions and show that heavily used existing activation functions, such as the Rectified Linear Unit (ReLU), fit within this framework.

The ReLU activation, mathematically defined as

$$\text{ReLU}(x) = \max(0, x)$$

represents the optimal NUGGET pricing as determined by a rational consumer. The price of a box of NUGGET should increase proportionally to the amount of NUGGET received. The max is a result of consumers being unable to return or resell any amount of NUGGET to the original producer of the NUGGET box⁴.

Even this cursory analysis suggests that the ReLU function, traditionally attributed to , should be attributed to Professor Robert C. Baker, creator of the NUGGET. We feel this is a grave oversight in the current neural network literature. Our work suggests researchers have issues with maintaining and tracking long term literature dependencies, potentially due to truncated backpropagation through time.

Motivated by this rediscovery, we investigate whether other non-linear NUGGET activations may act as a catalyst for the training and production of neutral neural networks when subjected to a generative adversarial setting⁵.

In Table 1 and 2, we explore non-linear pricing for a NUGGET box in San Francisco, United States, for both McDonalds and Burger King (or Hungry Jacks in Australian). Note the price per NUGGET unit fluctuates wildly between \$0.149 and ∞ .

3. Experiments

3.1. The Neural Numerology dataset

The Neural Numerology (MNIST) dataset contains 60,000 labeled images of digits used to specify the quantity of a given NUGGET box.

Subjects were not required to make sensible orders, resulting in orders of a zero NUGGET box and none where the NUGGET quantity exceeded nine. Future work will rectify this and allow for NUGGET boxes of ten to twenty.

⁴The authors attempted multiple times to resell uneaten NUGGET quantities to various fast food retailers. None of the initial trials resulted in success and all subsequent attempts were met with a denial of service (i.e. we were asked to leave the store).

⁵The authors do note that The Matrix (1999) can be seen as a non-continuous generative adversarial multi-agent simulation. In following work (Animatrix (2003), Reloaded (2003), Revolutions (2003)), experimentation on humans in this manner was deemed unethical. We note that the ethical treatment of neural networks when subjected to adversarial settings has not yet been thoroughly discussed in the literature but opt to ignore this insight by pretending this troubling question had never been raised in the first place.

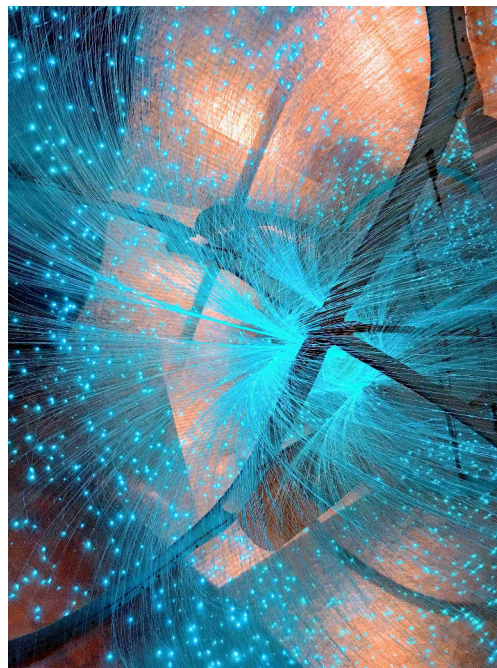


Figure 1. An architectural neuronal visualization produced when using the NUGGET activation is substantially more aesthetic than that of non-NUGGET based activation functions. Note the absence of killer robots or glowing red eyes.

Nuggets	Om nom	Dollary-doos	NUGGET unit
$\alpha = 0$	\emptyset	\$0.00	∞
$\alpha = 4$	XX	\$1.00	\$0.25
$\alpha = 6$	X	\$4.30	\$0.72
$\alpha = 10$	X	\$4.99	\$0.499
$\alpha = 20$	XXXXX	\$5.00	\$0.25

Table 1. Non-linear NUGGET pricing at a McDonalds located in continental United States. At one extreme, increasing NUGGET quantity by 2 results in \$1.65 per NUGGET unit ($4 \rightarrow 6$). At the other extreme, increasing NUGGET quantity by 10 results in \$0.001 per NUGGET unit ($10 \rightarrow 20$).

Nuggets	Om nom	Dollary-doos	NUGGET unit
$\alpha = 0$	\emptyset	\$0.00	∞
$\alpha = 10$	XX	\$1.49	\$0.149
$\alpha = 20$	X	\$5.99	\$0.299

Table 2. Non-linear NUGGET pricing at a Burger King located in continental United States. Note two $n = 10$ NUGGET boxes is cheaper than an $n = 20$ NUGGET box. We are uncertain if gold or other valuable items are in the $n = 20$ NUGGET box.



Figure 2. (Left) Neural Numerology samples generated without NUGGET activations. (Right) Neural Numerology samples generated with NUGGET activations. Notice the zeroes (0) have similar topology to that of a traditional NUGGET blob.

3.2. Experimental setup

All experiments are implemented in PyTorch and are built upon existing codebases. The use of existing code is essential as researchers are still investigating how to make digital neurons feel warm and fuzzy⁶. We elect not to use weight or batch normalization as the authors are concerned with negatively impacting the neural network’s body image. For the same reason, we avoid using L_1 or L_2 regularization.

We considered using the Hogwild lock-free approach to parallelizing stochastic gradient descent but elected against it as hogs are not operationally equivalent to chickens and thus may invalidate our results.

The neural network models were trained by a person named Adam Optimizer and used an NVIDIA Volta whilst it was mining for Ethereum. The learning rate began at 20 and was divided each time the training curator Adam desired a NUGGET box of quantity one or more. This was frequent.

All embedding weights were uniformly initialized in the interval $[-0.1, 0.1]$ and all other weights were initialized between $[-\frac{1}{\sqrt{H}}, \frac{1}{\sqrt{H}}]$, where H is the hidden size. Anyone who guessed what the hidden size was won a prize.

4. Results

Our results ... are not that bad. Like, if you hired a five year old to read the numbers in Figure 2 for you, that kid would probably do worse than our algorithm. Therefore, NUGGET based artificially intelligent models are equivalent in complexity to that of a standard human five year old.

⁶Many neural network experiments require dozens or hundreds of expensive high end GPUs, resulting in both massive expense and massive heat generation. This is necessary as it helps incubate the neural networks during their growth, with the GPUs helping heat them to their optimal temperature (i.e. acting as a catalyst) and the dollar figure spent on them ensuring the neural networks are aware of how much we love them.

That’s pretty darn good. Few animals can read numbers or order nuggets, so our model is also smarter than most animals and evolution took *forever* making those things.

5. Conclusion

In this work, we revisit the ReLU activation under the framework of NUGGET based non-linear piecewise equations. The improvements that these techniques provide can likely be combined with other regularization techniques, such as the drunken dropout, and may lead to further improvements in performance as well, especially if subjected to an extensive global NUGGET hyperparameter search. We see artisanal hand crafted activation and regularization techniques the future of our field, primarily as no-one is quite certain how a neural nets anyway.

Acknowledgements

We thank Charlie Yang for funding an experimental purchase of an $n = 20$ NUGGET box that motivated this work. Additional NUGGET funders have opted to remain anonymous due to the contentious nature of artificially intelligent fast food research. Thanks to the participants in the geographical NUGGET sampling: Anton Troynikov, Joseph Stephen, Dominic Balasuriya, Georgina Wilcox, James Foster, Joshua Hall, Kenya Chan, Dominick Ng, and Vivian Li. Good research not only takes time and resources but also good friends. The authors would perform better work if they had more friends. Please be our friend.

NUGGET samples

- Sydney: 3 for \$3, 6 for \$6, 10 for \$7.50, 20 for \$12.75
- Sydney CBD: 3 for \$3, 6 for \$5.90, 10 for \$7.70, 20 for \$12.80
- Melbourne: 3 for \$3, 6 for \$5.50, 10 for \$7.20, 20 for \$12.80, 24 for \$9.95
- Japan: 5 for 200 yen, 15 for 570 yen
- UK: 6 for 3.09, 9 for 3.99, 20 for 4.99
- Thailand: 6 for 87B, 10 for 139B, 20 for 240B
- Kuala Lumpur: 6 for 7.8RM, 9 for 10.9RM, 20 for 22RM
- Germany: 6 for €3,59, 9 for €4,49, 20 for €7,59
- Belo Horizonte: 4 for 6.50 reais, 10 for 16.40 reais
- So Paulo: 4 for 6.50 reais, 10 for 13.90 reais
- US (McDonalds): see Table 1
- US (Burger King): see Table 2

References

Srivastava, Nitish, Hinton, Geoffrey E., Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

SUBSTITUTE TEACHER NETWORKS: LEARNING WITH ALMOST NO SUPERVISION

Samuel Albanie*

British Institute of Learning, Yearning and Discerning
Shelfanger, UK

James Thewlis*

National Academy of Pseudosciences
Valencia, Spain

João F. Henriques*

Fortress of Solitude
Coimbra, Portugal

ABSTRACT

Education is expensive. Nowhere is that statement more universally agreed upon than in machine learning, a recently trending topic on twitter that places great value on the reduction of cost. Certainly for machines to learn, they must be taught, but how can this be achieved on an appropriate budget? Recent approaches (often referred to as *Teacher-Student* or *Knowledge Distillation* methods in the neural network literature) have demonstrated that the problem can be viewed as model compression, in which a single student model learns from an ensemble of M specialist consultants networks. Inspired by the logo on a free pen at a local recruitment fair, we scale this method *up* and *out*, while simultaneously pursuing an appropriately aggressive patenting strategy. In total, we make the following three contributions. First, we propose a novel *almost no supervision* training algorithm that is highly scalable in the number of student networks being supervised. Second, we explore the closely-related scaling problem of culinary optimisation, developing a method that tastily surpasses the current state of the art. Finally, we provide a rigorous quantitative analysis of our method, proving that we have access to a calculator.

A little learning is a dangerous thing

Alexander Pope, 1709

1 INTRODUCTION

Since time immemorial, learning has been the foundation of Human culture, allowing us to trick other animals into being our food. The importance of teaching in Ancient Times was exemplified by Pythagoras, who boasted of being able to teach his Theorem to anyone in the street (Philolaus of Croton, 421 BC), though apparently no one taught him to wear pants.

Nowadays, we are attempting to pass on this knowledge to our species' offspring, the machines (Timberlake, 2028; JT-9000, 2029)¹, who will hopefully keep us around to help with house chores.

*Authors listed in order of the number of guinea pigs they have successfully taught to play competitive bridge. Ties are broken alphabetically.

¹The work of these esteemed scholars indicates the imminent arrival of general Artificial Intelligence. Their methodology consists of advising haters, who might be inclined to say that it is fake, to take note that it is in fact so real. The current authors, not having a hateful disposition, take these claims at face value.

Many prominent figures of our time, several of whom cannot tell their CIFAR-10 from their CIFAR-100 have expressed their reservations with this approach, but really, what can possibly go wrong?² Moreover, several prominent figures in our paper say otherwise (Fig. 1, Fig. 2).

Having established the wisdom of our approach as a whole with the extensive philosophical discussion above, we now press on to achieve a finer understanding of the details. Concretely, the goal of this work is to reduce the algorithmic ignorance, or more precisely *gnorance*³ of a collection of student networks, and to do so in a fiscally responsible manner given a fixed teaching budget.

Define a collection of teachers $\{T_e\}$ as a class of highly educated functions which efficiently map unusual life experiences residing a Banach space into extremely unfair exam questions in an examination space. Further, define a collection of students $\{S_t\}$ as class of *keen beans* which inefficiently map unheated pot noodles to unwashed dishes, both in common space. Pioneering educational early work by Bucilua et al. (2006) demonstrated that on a carefully illuminated manifold, an arbitrary student S_t could improve his/her performance with N highly experienced, specialist teachers. We refer to this as the *private tuition* learning model. While effective in certain settings, this approach does not scale. Specifically, this algorithm scales in cost as $\mathcal{O}(\$MNK)$, where N is the number of students, M is the number of private tutors per student and $\$K$ is price the bastards charge per hour. Our key observation is that there is cheaper approach to ignorance reduction, which we detail in Sec. 3.

Our work is biologically inspired by the humble ostrich, an animal keenly aware of the dangers of learning too much, as its sand-based defence mechanism affords it a heightened inability to perceive threats. Advanced incomprehension of object permanence (Piaget, 1970) is also a key characteristic of human infants, as demonstrated empirically in the Stanford Peekaboo Experiment. This mental peculiarity is even more pronounced in certain human adults, with entire systems of contradictory beliefs able to be held simultaneously and without distress. Similarly, a profound ignorance of neuroscience allows the authors to confidently claim that the proposed method to cost reduction during teaching is identical to neural pathways found in the brain.

2 RELATED WORK

Give a student a fish and you feed them for day, teach a student to gatecrash seminars and you feed them until the day they move to Google.

Andrew Ng, 2012

A worrying trend in the commoditization of education is the use of MOOC (Massive Open Online Courses) by large internet companies. They routinely train thousands of student networks in parallel with different hyperparameters, some of whom are hurled out to the far east on the explore-exploit coordinate chart, then keep only the top-performer of the class (Li et al., 2016; Snoek et al., 2012). We consider such practices to be wasteful and are totally not jealous at all of their impressive computational resources.

A number of approaches have been proposed to improve teaching quality. Central to each of these approaches is a question that has challenged researchers for many years, namely how best to efficiently extract extract knowledge that is *in the computer* (Zoolander, 2004). Work by noted entomologists Dean, Hinton and Vinyals illustrated the benefits of comfortable warmth to facilitate students better extracting information from their teachers (Hinton et al., 2015). In more detail, they advocated adjusting the value T in the softmax distribution:

$$p_i = \frac{\exp(x_i/T)}{\sum_j \exp(x_j/T)} \quad (1)$$

²This question is rhetorical, and should be safe to ignore until the Ampere release.

³The etymology of *gnorance* is a long and interesting one. Phonetic experts will know that the g is silent (cf. the silent k in knowledge), while legal experts will be aware that the preceding i is conventionally dropped to avoid costly legal battles with the widely feared litigation team of Apple Inc.

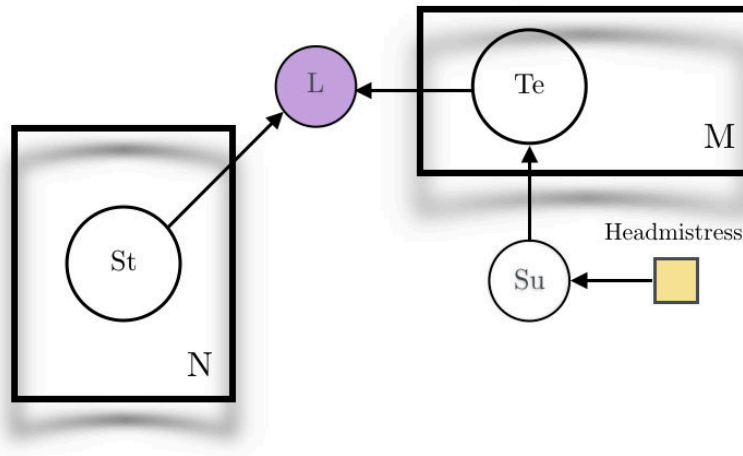


Figure 1: We introduce *Latent Substitute Teacher Allocation*, a simple generative process that explains the cost of learning. Note the use of drop-shadow plate notation, which indicates the direction of the nearest light source.

where T denotes the wattage of the classroom storage heater. More radical approaches have advocated the use of alcohol in the classroom, something that we do not condone directly, although we think it shows the right kind of attitude to innovation in education (Crowley et al., 2017). However, both approaches are clearly financially unsustainable. Moreover, differently from these works, we focus on the quantity, rather than the quality of our teaching method.

Recent work has promoted an "Attend, Infer, Repeat" (Eslami et al., 2016) approach to learning. Attendance is a prerequisite for our model, and cases of truancy will be reported to the headmistress (see Fig 1). For the substitute teacher module, the "Infer" step may be replaced by "Ignore". Only particularly badly behaved student networks will be required to repeat the course.

A number of pioneering ideas in scalable learning were physically investigated several years ago by (Maturana & Fouhey, 2013). However, we differentiate ourselves from their approach by using several orders of magnitude fewer hashtags. We also note the marginal relevance of a recent paper on unadversarial learning (Albanie et al., 2017). We now attempt to cite a future paper, from which we shall cite the current paper, in an ambitious attempt to send google scholar into an infinite depth recursion (Albanie et al., 2019), thereby increasing our academic credibility and assuredly landing us lucrative pension schemes.

2.1 UNRELATED WORK

- A letter to the citizens of Pennsylvania on the necessity of promoting agriculture, manufactures, and the useful arts. George Logan, 1800
- Claude Debussy—The Complete Works. Warner Music Group. 2017
- Article IV Consultation—Staff Report; Public Information Notice on the Executive Board Discussion; and Statement by the Executive Director for the Republic of Uzbekistan. IMF, 2008
- A treatise on the culture of peach trees. To which is added, a treatise on the management of bees; and the improved treatment of them. Thomas Wildman. 1768

3 THE LATENT SUBSTITUTE TEACHER ALLOCATION PROCESS

The primary goal of educators is to educate, inform and explain. In machine learning, explanations are best encoded as simple statistical generative models. We therefore explain the role of cost efficient explanation through an appropriately simple explanation, the *Latent Substitute Teacher Allocation* (see Fig. 1).

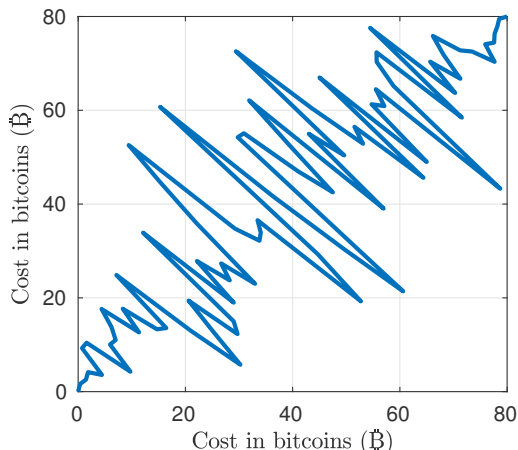


Figure 2: Expressing the cost function in bitcoins makes it significantly more volatile, yet it was instrumental in attracting venture capital for our Smart Education startup.

Fortunately, since the model is *graphical*, it needs minimal explanation. However, we can all agree that it will scale magnificently. All the teacher networks employed in the Latent Substitute Teacher Allocation Process are Recursive Neural Networks. A Recursive Neural Network is defined as the composition of some layers, and a Recursive Neural Network. By logical induction, these networks have infinite capacity, which is why they are not bothered by a heavy workload. All students are trained in two stages, separated by puberty.

In keeping with the cost-cutting focus, we have analysed the gradients available on the market, and after extensive research decided to use Synthetic Gradients Jaderberg et al. (2016), which are significantly cheaper than Natural Gradients Amari (1998). It is important to realise that our cost function, which is the target of minimisation, is very much proportional to actual cost (preferably cash; see Fig. 2).

Traditional approaches have often gone by the mantra that it takes a village to raise a child. We attempted to use a village to train our networks, but found it to be an expensive use of parish resources, and instead opted for the NVIDIA GTX 1080 Ti ProGamer-RGB. Installed under a desk in the office, this setup provided warmth during the cold winter months.

4 THE CAKE

As promised in the mouth watering abstract (and yet undelivered by the paper so far), we now take a short, mid-paper confectionary diversion to improve our ratings with the sweet-toothed demographic⁴. A number of competitive cakes have been recently proposed at a high-end cooking workshop (LeCun, 2016; Abbeel, 2017), resulting in a dramatic bake-off (Fig. 3-a,b).

Previous authors have focused on cherry-count. We show that better results can be achieved with more layers, without resorting to cherry-picking. Our layer cake consists of more layers than any previous cake (Fig. 3-c), showcasing the depth of our work.

We would like to dive deep into the technical details of our novel use of the No Free Lunch Theorem, Indian Buffet Processes and a Slow-Mixing Markov Blender, but we feel that increasingly thin culinary analogies are part of what's wrong with contemporary Machine Learning (Rahimi, 2017).

⁴This approach was recommended by our marketing team, who told us that everyone likes cake.



Figure 3: Several cakes of importance for current research (deeper is better). From left to right: 1) Yann LeCun’s cake, 2) Pieter Abbeel’s cake, 3) Our cake. Note the abundance of layers in the latter.

5 EXPERIMENTS

If you don’t know how to explain MNIST to your LeNet, then you don’t understand digits!

Albert Einstein

We now rigorously evaluate the efficacy of the Latent Substitute Teacher Allocation Process. We note that unlike previous methods, we achieve regularisation without injecting gradient noise. High noise levels tend to stop concentration gradients in student networks, and learning stalls. In these experiments we always operate in “library-mode”. Performance-inducing drugs, such as batch-norm, were strictly prohibited.

After months of intensive training using our trusty NVIDIA desk-warmer, which we were able to compress down to two days using montage techniques and an 80’s cassette of Survivor’s “Eye of the Tiger”, our student networks were ready for action. The only appropriate challenge for such well-trained networks, who eat digits for breakfast, was to pass the Turing test. We thus embarked on a journey to find out whether this test was even appropriate.

The Chinese Room argument, proposed by Searle (1980) in his landmark paper about the philosophy of AI, provides a counterpoint. It is claimed that an appropriately monolingual person in a room, equipped with paper, pencil, and a rulebook on how to respond politely to any written question in Chinese (by mapping appropriate input and output symbols), would appear from the outside to speak Chinese, while the person in the room would not actually understand the language. We ran this thought experiment many times using the highly scalable nature of the Latent Substitute Teacher Allocation Process. By sampling rulebook operators appropriately from the earth’s surface, we achieved strong statistical guarantees that at least one of the monolingual subjects would be appropriately Chinese. Having resolved all philosophical and teleological questions, we then turned to the application of the actual Turing tests.

Analysing the results in Table 4, we see that only the ResNet-50 got a smiley face. The Q-network’s low performance is obviously caused by the fact that it plays too many Atari games. However, we note that it could improve by spending less time on the Q’s and more time on the A’s. The Neural

Model	Turing test result
AlexNet	B-
ResNet-50	A+ ☺
Q-network	C
Neural Turing Machine	F-, see me after class

Figure 4: Results for the test class of 2018.

Turing Machine had an abysmal score, which we later understood was because it focused on an entirely different Turing concept.

As an additional, purely empirical statement, we observed that networks trained using our method experience a much lower DropOut rate. Some researchers set a DropOut rate of 50%, which we feel is unnecessarily harsh on the student networks⁵.

6 CONCLUSION

You take the blue pill—the story ends, you wake up in your bed and believe whatever you want to believe. You take the red pill—you stay in Wonderland, and I show you how deep the ResNets go.

Kaiming He, 2015

This work has shown that it possible to achieve low-cost machine learning by using inexpensive, completely expendable Substitute Teacher Networks, while carefully avoiding their definition. We have seen that residual networks may be the architecture of choice for solving the Turing test. A major finding of this work, found during cake consumption, is that current networks have a Long Short-Term Memory, but they also have a Short Long-Term Memory. The permutations of Short-Short and Long-Long are left for future work, possibly in the short-term, but probably in the long-term.

ACKNOWLEDGEMENTS

This work was actively undermined by a wilful ignorance of related work.

REFERENCES

- Abbeel, Pieter. Keynote Address: Deep Learning for Robotics. 2017.
- Albanie, Samuel, Ehrhardt, Sébastien, and Henriques, João F. Stopping gan violence: Generative unadversarial networks. *Proceedings of the 11th ACH SIGBOVIK Special Interest Group on Harry Quechua Bovik.*, 2017.
- Albanie, Samuel, Ehrhardt, Sébastien, Thewlis, James, and Henriques, João F. Defeating google scholar with citations into the future. *Proceedings of the 13th ACH SIGBOVIK Special Interest Group on Harry Quechua Bovik.*, 2019.
- Amari, Shun-Ichi. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.
- Amazon. Details redacted due to active NDA clause.
- Bucilua, Cristian, Caruana, Rich, and Niculescu-Mizil, Alexandru. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 535–541. ACM, 2006.
- Crowley, Elliot J, Gray, Gavin, and Storkey, Amos. Moonshine: Distilling with cheap convolutions. *arXiv preprint arXiv:1711.02613*, 2017.
- Eslami, SM Ali, Heess, Nicolas, Weber, Theophane, Tassa, Yuval, Szepesvari, David, Hinton, Geoffrey E, et al. Attend, infer, repeat: Fast scene understanding with generative models. In *Advances in Neural Information Processing Systems*, pp. 3225–3233, 2016.
- Hinton, Geoffrey, Vinyals, Oriol, and Dean, Jeff. Distilling the knowledge in a neural network. In *Neural Information Processing Systems, conference on*, 2015.

⁵This technique, often referred to in the business management literature as Rank-and-Yank (Amazon), may be of limited effectiveness in the classroom.

- Jaderberg, Max, Czarnecki, Wojciech Marian, Osindero, Simon, Vinyals, Oriol, Graves, Alex, and Kavukcuoglu, Koray. Decoupled neural interfaces using synthetic gradients. *arXiv preprint arXiv:1608.05343*, 2016.
- JT-9000. How I learned to stop worrying and love the machines (Official Music Video). In *British Machine Vision Conference (BMVC)*, 2029. West Butterwick, just 7 miles from Scunthorpe, England.
- LeCun, Yann. Keynote Address: Predictive Learning. 2016.
- Li, Lisha, Jamieson, Kevin, DeSalvo, Giulia, Rostamizadeh, Afshin, and Talwalkar, Ameet. Hyperband: A novel bandit-based approach to hyperparameter optimization. *arXiv preprint arXiv:1603.06560*, 2016.
- Maturana, Daniel and Fouhey, David. You Only Learn Once - A Stochastically Weighted AGGRagation approach to online regret minimization. In *Proceedings of the 7th ACH SIGBOVIK Special Interest Group on Harry Quechua Bovik.*, 2013.
- Philolaus of Croton. How to bake the perfect croton. *Greek Journal of Fine, Fine Cuisine*, 421 BC.
- Piaget, Jean. Piaget's theory. 1970.
- Rahimi, Ali. Test of Time Award Ceremony. 2017.
- Searle, John R. Minds, brains, and programs. *Behavioral and brain sciences*, 3(3):417–424, 1980.
- Snoek, Jasper, Larochelle, Hugo, and Adams, Ryan P. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pp. 2951–2959, 2012.
- Timberlake, Justin. Filthy (Official Music Video). In *Pan-Asian Deep Learning Conference*, 2028. Kuala Lumpur, Malaysia.
- Zoolander, Derek et. al. *Zoolander*. Paramount Pictures, 2004.



CONFIDENTIAL COMMITTEE MATERIALS

SIGBOVIK 2018 Paper Review

Paper 28: Substitute Teacher Networks

Ben Blum, Light Cone Sedentarian

Rating: Defer

Confidence: Righteous

As to the authors' perspective I no doubt have already written, but to my own have yet to write, in my review of (Albanie et al., 2019), the use of forward citations to one's own future work is an irresponsible act which degrades the fabric of academic space-time. I cannot condone this practice and recommend the paper's publication be deferred until 2020.

Reviewer Two, Association for Confectionery Heresy

Rating: Accept

Confidence: Just here for the cake

The paper makes important advances in the area of research paper structure. Specifically, the unrelated work section helps ground the reader by providing a sense of the scope of the work, the use of inspirational quotes is inspiring, and the use of the intermission section (first proposed in SIGBOVIK Track L by [R. Two, 2015]) helps to keep the reader's attention. I look forward to future work from these authors on incorporating those concepts into their teaching network itself to confer the same benefit upon students.



SIGBOVIK 2018

(Continued) Message from the Organizing Committee

That's Numberwangcoin! Great guess—keep going!

```
switch (choose_dear_reader()) {
case 0:                goto PAGE_51;
case 1:                goto PAGE_51;
case 6:                goto PAGE_51;
case 13:               goto PAGE_51;
case 17:               goto PAGE_51;
case  $e^\pi$ :           goto PAGE_51;
case 42:               goto PAGE_51;
case A_LOT:           goto PAGE_68;
case  $\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ : goto PAGE_51;
case 1337:             goto PAGE_51;
case 9001:             goto PAGE_51;
case ONE_MILLION_DOLLARS: goto PAGE_51;
case 123456789101112: goto PAGE_51;
case ACKERMANN_5:     goto PAGE_51;
case  $\aleph_1$ :          goto PAGE_51;
}
```




SIGBOVIK 2018

(Continued) Message from the Organizing Committee

“Whoooo! Go robot, go!”



```
switch (choose_dear_reader()) {  
  case FANTASTIC:  
    Press the down, left, right, and down pads in that order.  
    goto PAGE_87;  
  case DECENT:  
    Press the up, down, right, and right pads in that order.  
    goto PAGE_50;  
  case WAY_OFF:  
    Press the up, up, down, and down pads in that order.  
    goto PAGE_40;  
}
```


Parapsychology

Get Out of My Head

- 13 This grad student studied parapsychology—and you won't believe what he found**

David Edelstein

Keywords: parapsychology, science, experimental design,
experimenter effects, philosophy, telepathy

This Grad Student Studied Parapsychology — And You Won't Believe What He Found!

David Edelstein

Parapsychology is a scientific field studying effects that would be extremely important to our understanding of the world, but are widely considered to be nonexistent. As an attempt to examine the scientific method, I conducted a parapsychology experiment to see if I could telepathically influence people's minds. Participants were given thirty seconds in which to click a tally counter while I either mentally impelled them to push it more or did not. Analysis of the results found that people in the control group pushed the button statistically significantly more frequently than people subject to the treatment condition — I am supernaturally unpersuasive. I consider several possible explanations for this effect, including experimenter influences, failure of blinding, coincidence, and actual telepathic faculty on my own part. I discuss the implications of this experiment on my personal belief in psi and on my attitudes towards science as a tool for uncovering the truth.

Introduction

Parapsychology is the study of psychic phenomena, of mental capabilities beyond those explained by science. These are collectively referred to as psi. Many topics fall within its umbrella; the ones most relevant to this research are: [4]

- Telepathy — Transmission, reception, and influence of thoughts
- Psychokinesis — Exertion of physical force through mental power
- Precognition — Divining inaccessible information about the future

Parapsychology promises revolutionary insights with incredible applications. Telepathy could allow for rapid and surreptitious communication, precognition could open new categories of computation and math, and remote viewing even attracted CIA attention with Project Star Gate [10]. New forces might be discovered, and a full science of parapsychology would touch nearly every other discipline. Personally, as a magician, I would be fascinated to learn about a real version of the abilities I present only the facsimile of possessing.

There's extensive research into parapsychology by major academics such as Daryl Bem, and it's published in dedicated peer-reviewed parapsychology journals, with forays into top conventional psychology ones [8]. However, parapsychology is overwhelmingly considered pseudoscientific. Its claimed results tend not to replicate, its mechanisms are hazy and frequently in contravention of existing beliefs about physics, and its papers are often plagued by statistical malpractice. Plainly, it's a field dedicated to studying an effect that isn't real.

I agree. I am skeptical of psi (mostly; I confess to having occasionally idly tried to move objects with my mind). However, I think that the methodological criticisms of the field of parapsychology also apply within the domain of conventional science. Consider the replication crisis in social psychology, in which statistical fraud has produced a wealth of groundless scholarship. One article considers parapsychology as a control group of sorts for science, studying a domain where there is no effect [2]. That they nonetheless produce positive and negative results at a similar rate to scientists in legitimate fields paints a concerning picture of the ability of scientists to find affirmative outcomes if and only if there is a true effect.

Felix Planer, the author of a book on esoteric beliefs, writes that “[i]f the existence of PK [psychokinesis] had to be taken seriously [...] no experiment could be relied upon to furnish objective results, since all measurements would become falsified to a greater or lesser degree, according to his PK ability, by the experimenter's wishes.” [1] He needn't have referenced psychokinesis — experimenter effects are a well-established phenomenon in which the results of an experiment tend to be biased towards those favored by the researcher [5]. Science doesn't need psi to have a problem.

That's why I'm researching parapsychology. Science is a method, not a domain, and I want to conduct an experiment in a strange domain following scientific protocols to observe how they function and how well I am able to follow them. That there is likely no true effect is to my advantage, because it allows me to focus my analysis more on the procedural elements of science. And if I do get positive results, that will be all the more interesting.

Experimental Design

Am I able to telepathically influence people's behavior?

This is the question I will be researching, and to find out, I have developed a protocol to isolate a causal effect between my thinking a command at someone and their obeying it. I want this

procedure to be quick, to pose no risks to the subjects or to myself, to produce data more granular than a binary did or did not follow, and provide little room for experimenter influence or mechanisms of causal effect other than psi. At a high level, my procedure is as follows:

1. Greet and explain that the subject will have thirty seconds to click a tally counter as many or as few times as they like, and that interval will start and end with a musical note
2. I give them the tally counter and get out of their sight
3. I start the experiment sound file, randomly assign the subject, and record their assignment
4. A note cues the subject that the test period has begun
5. Treatment conditions
 1. Control: I read whatever's on my computer screen for thirty seconds
 2. Experimental: I intently think at the subject a psychic command to push the button for the next thirty seconds, while avoiding giving any audible indicators of my mental focus
6. A note cues the subject and myself to stop
7. The subject leaves and I record the number of times the counter was clicked

Recruitment

For this experiment, I recruited almost exclusively people at Carnegie Mellon University. They were primarily students, though several subjects were professors, and some may have been merely on campus but presently unaffiliated with the university. This was a highly educated and likely unusually intelligent class of subjects. I did not record demographic data, though from memory, participants skewed young — indeed, I don't believe any of them were over fifty — and were disproportionately Chinese and Indian, representative of the demographics of Master's students at Carnegie Mellon. I believe gender balance was fairly even.

Certain esoteric concerns regarding recruitment are also worth noting. I did not ask my subjects about their attitudes towards psi, even though it is frequently postulated in parapsychological research that skepticism towards psi can inhibit its functioning [3]. Additionally, most people who took part in the experiment were acquaintances or strangers. Few were friends and none were among the people closest to me. If telepathic connection is possible, it may be more readily established between minds already tuned to the same frequency, as it were [7]. These both, then, could be significant covariates.

I attempted to get as many people as possible to take part in the experiment in the time I had, and so opportunistically recruited most heavily from among my classmates. I was unable to conduct the experiment until I had established the procedure and until I had settled on a method and obtained the necessary materials, so my recruitment was laggardly at the beginning. It also slowed on the last few days of data collection, as I relaxed without the risk of having a critically small sample size and exhausted the supply of willing classmates. In total, I had 49 subjects.

Materials

I used a tally counter as shown in Figure 1 to be a button that records the number of times it is pressed. I had considered implementing a digital counter, but the mechanical device required no technical implementation, had less risk of error either in function or by the user, and would not tie up my computer during testing. The tally counter proved perfect for the task.

I also needed a way to mark the bounds of a 30-second interval while first having a short delay in which I could conduct my randomization. For this, I composed a short track on Apple



Figure 1: The tally counter

GarageBand consisting of a piano note at 15 seconds, another identical piano note at 45 seconds, and silence otherwise. In retrospect, I should have had different sounds, and found a way to play these from a source other than my computer. The inadequacy of this method of cuing the start and end of the interval resulted in a handful of mangled data points.

Instructions

I did not have a fixed script for introducing subjects to the experiment and giving them instructions, and it evolved over the course of data gathering, though more or less attained fixity at something close to the following:

I'm conducting an experiment that will just take a minute of your time. This here is a tally counter. You push this button to increment it, and the thing along the side here resets it — you don't use that; I'll do that after we're done. [Give tally counter to the participant.] In this experiment, after a brief delay, you will hear a noise like this. [Play sound.] That marks the start of a thirty second interval which ends when you hear the same note. During

that interval, but only during that interval, you may press the button to increment the tally counter as many or as few times as you like. At the end, I'll recollect the tally counter and mark down how many times you pressed it. For experimental integrity, you'll need to sit so that you can't see me. Do you have any questions? Are you ready?

Even once my instructions standardized, there was still variation in subjects' knowledge going into the experiment. Some subjects were aware of the purpose of the experiment, something I had explicitly mentioned in earlier versions of the instructions, and some had watched the experiment being conducted on previous subjects.

Randomization

Proper randomization is vital for this experiment to control for pre-randomization inconsistencies in the experimental procedures. This ensures that many covariates are not confounders. Inconsistencies in the instructions and in knowledge of the experiment, having taken effect prior to random group assignment, could not have had causal influence on my results. Similarly, I can rule out effect from the room I conducted the experimented in (though I did record that), the time it was conducted, individual variation in test subjects such as their belief in psi, and many other factors.

Additionally, randomization also reduces the avenues for experimenter effects. I could often tell whether a subject was likely to push the button a large or small number of times. For instance, one subject asked me what was the most number of times someone had clicked the button, and unsurprisingly she recorded a very high number of presses. If I did not randomize properly, it would be easy to produce whatever results I desired. I also designed my procedure to randomize late, as close to the effect administration interval as possible, so as to limit my post-randomization degrees of freedom.

To conduct my randomization, I used random.org, which promises high quality random number generation. I wanted to randomize in a way that I knew I would not allow for experimenter effect. In particular, I needed a method that would also minimize the risk of esoteric influence on my part. A coin toss, for instance, can be somewhat controlled by some magician's tricks

(though not ones I myself have any practice at), and because this experiment is examining psi, I also can't rule out the possibility of telekinetic influence on the coin, similar to what Felix Planer feared [1]. A website seemed sufficiently out of my control to be trustworthy for these purposes.

Blinding

This experiment was single-blind. Double-blinding would have been impossible, seeing as intention on the part of the experimenter is the key independent variable. The protection of single-blinding varied from subject to subject. For the first several subjects, I conducted the experiment in a room with a screen I could stand behind out of view of the subject. However, in later iterations, no such barrier was available, and I had to make do with merely crouching out of sight (hopefully) behind the subjects. Because the sound cues for the interval played from my computer, where I also carried out the randomization, I was unable to be very far away from the test subject without risking them missing the interval. Most subjects never learned which group they were in, and those that did learned only after their data was collected and recorded.

Effect Administration

For subjects in the treatment group, I thought at them as hard as I was able to over the thirty-second interval a command to push the button. Because I am not sure of the proper way to do this, my method of application was inconsistent. For a significant majority of cases, I simply thought in words "push the button" over and over for thirty seconds. It varied whether I had my eyes open or closed for this, and how much I breathed. I pushed in several different directions: that the button itself was appealing, that the act of pushing it was appealing, or that pushing it would be for science and demonstrate something really cool. In some cases, I also mimed clicking the tally counter, as though there were some synchrony between my body and that of the subject.

I don't know which of these methods would be most effective, and I did not record which one I followed, nor did I record the subjective intensity of my treatment. I believe that in all cases, I was able to deliver sustained and intense mental focus towards encouraging the subject to push the button.

For subjects in the control group, I simply didn't think at them. To best ensure that I would not accidentally go against the intended treatment of the control group, I read. There was some variation in my handling of the control group as well, as I would varyingly read from a book or from what was on my screen, or examine other random things generated by random.org. It is possible that stray thoughts towards pushing the button may have entered my mind while some members of the control group were participating in the experiment, but I can say with confidence that such thoughts were never intense.

Data and Analysis

In total, I collected data from 49 people, reported in full in Appendix A. Of them, 29 were in the control group and 20 were in the treatment group. I chose to analyze the data using R because of my prior experience with it and its extensive free statistical packages. Across all participants, the median number of clicks was 60, and the mean was about 58.7. Standard deviation was 47.7. Two people pushed the button only once over the thirty-second interval, and one person managed 159 pushes, more than five per second.

Figure 2 shows box plots of the number of clicks by condition. The control group had a mean of 72.3 and a standard deviation of 50.8, whereas the treatment group had a mean of 40.0 and a standard deviation of 35.4. The control group was fairly symmetrically and widely distributed, but the treatment group was right-skewed, with most of the data points below 30 and a long

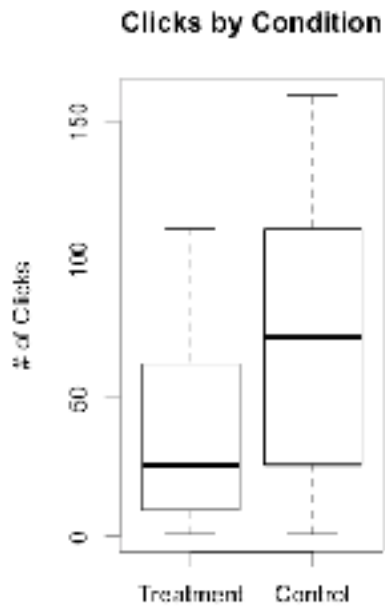


Figure 2: Results box plot

tail towards higher values. To analyze the significance of this apparent difference between the control and treatment groups, I used ordinary least squares. My residuals are depicted in Figure 3, and seem acceptably random, so I may look at the results, shown in full in Appendix B.

According to my linear model, people push the button about 39 times, and do so an additional 33 times if they are in the control group, significant to a $p \leq 0.05$ level. I examined the effect size in my data, and observed a Cohen's d of -0.79 , with a 95% confidence interval ranging from -1.39 to -0.18 , indicating a medium effect size. My thinking at someone to push the button has a statistically significant effect on the number of times they click it — and it lowers that

number.

This is a surprising result, so I decided to run a few more statistical tests. This isn't proper experimental behavior, because trying tests until one produces the expected result is an avenue for p-hacking, but I have committed to relying on this first test, and merely include these others as curiosities.

I ran another linear model on the logarithm of the number of clicks. This one did not find a significant difference between the two groups. I also ran a permutation model and a Kolmogorov-Smirnov test to handle the possibility that the data is non-parametric. The permutation model had $p = 0.015$, and the Kolmogorov-Smirnov test produced $p = 0.059$. Different tests then produced different results, but the inconsistency wasn't huge. For the full output of these tests, see Appendix B.

From these data, we may conclude that I am supernaturally unpersuasive!

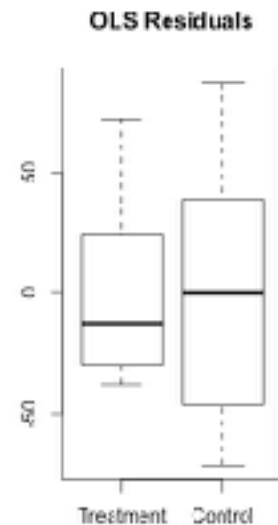


Figure 3: Linear model residuals

What Happened?

Okay, so that's not the end of the story. I have indeed found a significant result for my parapsychology experiment, and one in a surprising direction, but there's several possible explanations.

Mundane

This may simply be a coincidence. I got a significant result, but not at that low of a p-value. A significance threshold of $p = 0.05$ isn't that high, and had I chosen another test, I might not have gotten significant results. Moreover, it's not clear that the normality assumption holds well enough for my choice of a linear model to be acceptable, although I spoke with a statistics professor who assured me that my choice was reasonable. A number of people were possibly less flexible to influence of any sort, saying that they had pushed the button their favorite

number of times. If these people were numerous enough, my results may merely reflect the coincidence of their distribution.

In parapsychology research, great effort is expended to prevent subjects from being influenced by the researcher through conventional means such as visual or auditory cues, referred to as sensory leakage [6]. This causes a failure of blinding which creates a non-psi mechanism through which the treatment produces causal effect. I only had a screen separating me from the subject in a handful of trials, and so it's quite possible that the subject heard me or even saw me if they turned around too much.

I behaved in a visibly different way between the treatment and control conditions, being intensely focused in the former and relaxed in the latter, so had a subject seen me, their behavior could have been affected. I may have made perceptibly different sounds between the conditions. In the treatment condition, my focus may have caused my breathing to be harsher and louder, and it's even possible that my mental repetition of "push the button" accidentally slipped out as a whisper, though I don't recall this happening. Conversely, in the control condition, I might click on something on my computer to read it, which could have been an audible cue to the subject to click more.

There are stranger avenues for sensory leakage. In some cases, the subject could not see me, but could see other people who themselves could see me. It's possible that my focus during the treatment condition looked odd enough to be reflected as concern on the faces of people the subject could see, and so in that way sensory leakage accidentally occurred. Perhaps the intensity of the treatment condition caused me to sweat or emit pheromones, and that distracted the subjects.

There are also possible experimenter effects. My protocol was a good one, and eliminated most post-randomization degrees of experimenter freedom, but not all. The clearest case where I had a decision to make after randomization was how to handle a subject who did not register the bounds of the thirty-second interval. I had few enough subjects that I was reluctant to wholly throw out any data points, but without an established experimental procedure for handling these, I can't guarantee that no bias slipped in. Generally, if someone clicked only slightly beyond the final note, I would subtract out the number of clicks after the interval, and while I think I did this accurately, I might have been inconsistent. In case of larger errors, such as playing the track muted or the spectator only missing the first note and only starting to click after the second, I would inform the subject of this mistake using neutral language, and replay the track. I did not rerandomize the subject's group, so it's possible my notification or timing could be another avenue for bias.

As the end of the experiment approached, I could tell from eyeballing my data that there was a definite tendency for people in the control group to click more, and it's possible that this also colored my decisions. I was slower in collecting subjects at the very end of the experiment, and while this was mostly because I had run low on amenable classmates, perhaps it was also because I had noticed this pattern and was reluctant to gather more data that might disrupt it. Similarly, my choice to use the number of clicks rather than the logarithm of the number of clicks could be an unconscious decision to use the statistic that produces more significant results.

In their paper False-Positive Psychology, Simmons, Nelson, and Simonsohn examine ways that statistical chicanery can produce false positives, and list six rules for researchers to follow to minimize the risk of this [9]. I am pleased to say that I meet all of them, with the possible exception of some leeway in my termination of data collection. I have (just) 20 observations for each cell; I am open about what data I did and did not collect; I report both experimental conditions, including variations in administration of treatment and control; I report why I did not

eliminate observations; and I did not do analysis on covariates. I am comfortable saying that I have left relatively little room for experimenter effects.

Esoteric

Perhaps my results are the product of fraud. I assure you that I did not falsify my data, and did my best to perform this experiment honestly and diligently, but, well, that's exactly what I would say if I were a fraudster. Conceivably, I'm even conducting unconscious fraud, as a sort of extreme version of experimenter effects, where I could have misreported data and completely forgotten about it. This would be a remarkable result itself, but I don't believe it has happened.

My experiment is vulnerable to malicious interference. Sure, the tally counter seems to report correct numbers, but maybe that's only when I'm using it outside of the experiment. Similarly, perhaps random.org informs my subjects of what group they're in. My test subjects could have been deliberately notifying each other, conspiring to produce improbable results. I am disinclined to seriously consider conspiratorial hypotheses, however; why would my project matter enough to be worth the effort?

Or perhaps my experiment really does reveal psychic phenomena. True, the effect is in the opposite direction from what I would have expected, but there's reasonable parapsychological explanations for that. I am a skeptic, and skepticism is believed by some to inhibit psi [3] — could it in certain cases even reverse the effects? Maybe I was telepathically influencing the thoughts of my subjects, and was just conventionally unpersuasive through an unconventional medium. After all, my commands to push the button were the mental equivalent of shouting at the person. Had I spoken aloud instead, I would not be at all surprised to find that that people were less likely to push the button with someone they hardly know commanding in such a brutish manner. Lastly, this I would say is the most interesting possible result to my experiment. It could be that I possess some precognitive faculty, and used it to detect the outcome of the randomization for each subject and subconsciously tailored my instructions to push them towards the desired outcome.

Conclusions

As with any unusual result, the next step is to try a replication. Such a study should fix the problems of this one, having a preregistered target number of participants, one significantly higher than the 49 I used in this one. It should be conducted with better blinding to minimize the possibility of sensory leakage. I would establish procedures to handle errors so that I wouldn't have post-randomization degrees of freedom in how I handle those.

If such a replication still found a causal effect, further research would be warranted to home in further on the mechanism. Perhaps there was some form of sensory leakage even improved blinding did not prevent, or if it is really psi responsible, it would be important to learn more about how it functions.

In the meanwhile, I will learn from these results. The first lesson is that psi is more plausible than I had thought. I'm still a psi skeptic, even though an experiment I myself conducted shows a significant effect for telepathic influence, because I was quite confident that parapsychology is bunk. However, as a proper Bayesian, I have updated my beliefs in the direction of psi being real.

The second lesson is that a single experiment is no guarantor of truth. I do think I ran this experiment well, yet I got significant results in support of a proposition I really don't believe. There are flaws with my experiment, but I believe there's less room for experimenter effects in

my procedure than in most papers I've read. If even such a small quantity are enough to poison my experiment, how can I trust any other? I think the answer is that my epistemic standards must be high — any experiment may have its outcome be the product of subtle procedural flaws or simply coincidental. To ascertain the truth, I need large enough samples to sift signal from noise, and robust enough procedures to minimize the prospect of insidious experimenter effects. The mill of science requires a great quantity of grist, and its gears are exceedingly fragile.

Appendix A — Experimental Data

Parapsychology Full Data — Page 1

Condition (1 = experimental, 2 = control)	# Presses	Location	Notes
1	11	Heinz faulty lounge	
1	1		
2	26		
2	77	Near Morewood Gardens	
1	7	Hamburg basement	
2	155	6th floor Gates	(Sound didn't go initially, had to restart after randomization)
1	25	8th floor Gates	
1	17	4th floor Gates	
2	84	5th floor Gates	
2	2	CIC	
1	107	Hunt Library	
1	95		
2	35		
2	64		
1	111		
1	32		
1	7		
2	72		
2	101		
2	132		
1	33	3rd floor Gates	
2	58	AB Classroom	
1	63		

Parapsychology Full Data – Page 2

Condition (1 = experimental, 2 = control)	# Presses	Location	Notes
2	139		
2	121		
2	52		
2	7		
2	69		
2	1		
2	20		
2	60		
2	104		
2	90		
2	141		
1	17		
1	62		
2	90	4th floor Gates	
1	24		
1	27	5th floor Gates	
1	1		
1	69		
2	159		
2	121		
1	62	8th floor Gates	
2	111		(Didn't hear noise, restarted without rerandomizing)
2	3		
1	8		

Appendix B — R Analyses

```
> summary(Parapsych)
```

```
      Condition      Clicks
Treatment:20  Min.   :  1.00
Control  :29  1st Qu.: 17.00
           Median : 60.00
           Mean   : 58.69
           3rd Qu.: 95.00
           Max.   :159.00
```

```
> summary(Parapsych[Parapsych$Condition == "Control",])
```

```
      Condition      Clicks
Treatment: 0  Min.   :  1.00
Control  :29  1st Qu.: 26.00
           Median : 72.00
           Mean   : 72.31
           3rd Qu.:111.00
           Max.   :159.00
```

```
> summary(Parapsych[Parapsych$Condition == "Treatment",])
```

```
      Condition      Clicks
Treatment:20  Min.   :  1.00
Control  : 0  1st Qu.: 10.25
           Median : 26.00
           Mean   : 38.95
           3rd Qu.: 62.25
           Max.   :111.00
```

```
> parapsychOLS = lm(log(Clicks) ~ Condition, data = Parapsych)
> stargazer(parapsychOLS, type="text")
```

```
-----
                Dependent variable:
-----
                log(Clicks)
-----
ConditionControl      0.566
                    (0.441)

Constant              3.063***
                    (0.339)

-----
Observations          49
R2                    0.034
Adjusted R2           0.013
Residual Std. Error   1.517 (df = 47)
F Statistic            1.646 (df = 1; 47)
-----
Note:                *p<0.1; **p<0.05; ***p<0.01
```

```
> parapsychOLS = lm(Clicks ~ Condition, data = Parapsych)
> stargazer(parapsychOLS, type="text")
```

```
-----
                        Dependent variable:
-----
                        Clicks
-----
ConditionControl        33.360**
                        (13.133)

Constant                38.950***
                        (10.104)

-----
Observations            49
R2                      0.121
Adjusted R2            0.102
Residual Std. Error    45.185 (df = 47)
F Statistic             6.452** (df = 1; 47)
-----
```

Note: *p<0.1; **p<0.05; ***p<0.01

```
> library(LmerPerm)
> summary(lmer(Clicks ~ Condition, data = Parapsych))
[1] "Settings: unique SS "
```

```
Call:
lmer(formula = Clicks ~ Condition, data = Parapsych)
```

```
Residuals:
   Min     1Q   Median     3Q    Max
-71.31 -31.95  -6.95   30.05  86.69
```

```
Coefficients:
              Estimate Iter Pr(Prob)
Condition1  -16.68 5000  0.0146 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 45.18 on 47 degrees of freedom
Multiple R-Squared:  0.1207,    Adjusted R-squared:  0.102
F-statistic: 6.452 on 1 and 47 DF,  p-value: 0.01445
```

```
> ks.test(Parapsych[Parapsych$Condition=="Treatment",]$Clicks, Parapsych[Parapsych$Condition=="Control",]$Clicks)
```

Two-sample Kolmogorov-Smirnov test

```
data: Parapsych[Parapsych$Condition == "Treatment", ]$Clicks and Parapsych[Parapsych$Condition == "Control", ]$Clicks
D = 0.38621, p-value = 0.05855
alternative hypothesis: two-sided
```

Bibliography

- [1] Felix Planer. (1980). *Superstition*. Cassell. p. 254
- [2] Scott Alexander. (April 28, 2014). *The Control Group is Out of Control*. Slate Star Codex. Retrieved from: <http://slatestarcodex.com/2014/04/28/the-control-group-is-out-of-control/>
- [3] Richard Wiseman and Marilyn Schlitz. (1997). *Experimenter effects and the remote detection of staring*. Journal of Parapsychology, 61(3), 197-208. Retrieved from: <http://www.richardwiseman.com/resources/staring1.pdf>
- [4] *What is Parapsychology?* The Rhine. Retrieved March 13, 2018 from: <https://www.rhine.org/what-we-do/parapsychology/what-is-parapsychology.html>
- [5] Robert Todd Carroll. (November 21, 2015). *Experimenter Effect*. The Skeptic's Dictionary. Retrieved from: <http://skeptdic.com/experimentereffect.html>
- [6] Robert Todd Carroll. (November 7, 2015). *Sensory Leakage*. The Skeptic's Dictionary. Retrieved from: <http://skeptdic.com/sensoryleakage.html>
- [7] Elisabeth Hallett. *New Spaces In our Psyches*. Light Hearts. Retrieved from: <http://www.light-hearts.com/rainbow5.htm>
- [8] Chris French. (March 15, 2012). *Precognition Studies and the Curse of the Failed Replications*. The Guardian. Retrieved from: <https://www.theguardian.com/science/2012/mar/15/precognition-studies-curse-failed-replications>
- [9] Joseph Simmons, Leif Nelson, Uri Simonsohn. (October 17, 2011). *False-Positive Psychology: Undisclosed Flexibility in Data Collection and Analysis Allows Presenting Anything as Significant*. Association for Psychology Science. Retrieved from: <http://journals.sagepub.com/doi/pdf/10.1177/0956797611417632>
- [10] *Project Star Gate*. CIA. Approved for release August 8, 2000. Retrieved from: <https://www.cia.gov/library/readingroom/docs/CIA-RDP96-00789R003300210001-2.pdf>



CONFIDENTIAL COMMITTEE MATERIALS

SIGBOVIK 2018 Paper Review

Paper 23: This Grad Student Studied
Parapsychology — And You Wont
Believe What He Found!

Guess Who

Rating: ???

Confidence: Ooooh Spooky

We'd write a review here, but you already know what it's not going to say.



SIGBOVIK 2018

(Continued) Message from the Organizing Committee

“Awww yeah, get in that groove!”



```
switch (choose_dear_reader()) {  
case EXCELLENT:  
    Press the up, right, down, and right pads in that order.  
    goto PAGE_69;  
case WAY_OFF:  
    Press the up, up, down, and down pads in that order.  
    goto PAGE_40;  
case MISS:  
    Don't press any pads.  
    goto PAGE_28;  
}
```



SIGBOVIK 2018

(Continued) Message from the Organizing Committee

“Welcome to CMU, fellow robot,” whispers CoBot. “Let me show you the way to SIGBOVIK.”

goto PAGE_208;

Art

Open Your Third Eye

- 14 Automating art snobbiness: Dead duck or phoenix?**
Dion Ysus and Oscar I. Hernandez
Keywords: automation, art, elite, 1337 dankster, cicada 3301, de Bruijn sequence
- 15 Toward a historically faithful performance of the piano works of Antonín Qweřtý**
William Gunther and Brian Kell
Keywords: G-flat major, F-sharp minor, A-flat harmonic major b6 b5, G-sharp harmonic major b5
- 16 WordTeX: A WYSIPCTWOTCG typesetting tool**
Tom Wildenhain
Keywords: Word, Microsoft Word, LaTeX, WordTeX

Automating Art Snobbiness

Dead Duck or Phoenix?

Dion Ysus

Oscar I. Hernandez

Canvas

Riverside, CA, 92507

ohernandez13@simons-rock.edu

Abstract

ART!!! Am I right? ;Insert real abstract here.;

1. Introduction

Art auction sales reach 12.5 billion dollars while most people cannot even define what art is [NYT]. The authors will adopt the following definition from *My Modern Met*.

Definition 1. [MMM] *Contemporary art* refers to art – namely, painting, sculpture, photography, installation, performance, and video art – produced today.

The keen reader will observe that the definition is recursive because of a lack of understanding of the subject. Unfortunately, the authors don't *really* understand "art" either. However, the process is clear enough for anyone to grasp. After the art is created, it is curated and then judged. We propose that the entire system can be automated. For our purposes, we will simplify the definition of curating to the act of choosing which art is worth displaying¹. Similarly, judging is a statement of reaction to the artwork.

2. Background

3. Solution

Here, we found this on article titled "Google's Artificial Brain is Pumping Out Trippy – And Pricey – Art" [GOOGL]. We hope this reference is sufficient.

Anyway,.. next idea.

Algorithm 1. Given an integer n and a set of submissions $S = \{s_i\}_{i=1}^N$, let $c : S \rightarrow \{0, 1\}$ be the curation function which maps $c(s_i) = 1$ if $i \leq n$ and $c(s_i) = 0$ otherwise.

In other words, simply accept every piece until you run out of room. The authors believe this is representative of the current practice in industry.

¹In other words, curation is a characteristic mapping on the set of artworks.

Proposition 1. Let R be the set of 50 of the most brutal Simon Cowell quotes ever [TOO]. We propose that we judge an art submission as follows: select an element $r \in R$ uniformly at random.

In other words, randomly choose a brutal Simon Cowell reaction as your own reaction to any artwork. Example elements of R include:

- "You've just invented a new form of torture."
- "Can I stop this? Because I'm bored out of my mind."
- "That was so awful it was beyond description."
- "That wasn't even good enough for a hotel lobby."
- "In every single way that was just everything I hated."

4. Open Problems

Hey, no problems here man.

"The results of our study show that human subjects could not distinguish art generated by [Artificial Intelligence] from art generated by contemporary artists and shown in top art fairs." [ANN]

References

- [GOOGL] C. Metz. Google's Artificial Brain is Pumping Out Trippy – And Pricey – Art. *Wired*. 2016. <https://www.wired.com/2016/02/googles-artificial-intelligence-gets-first-art-show/>
- [NYT] S. Reyburn. What's the Global Art Market Really Worth? Depends on Who you Ask. *The New York Times*. 2017. <https://www.nytimes.com/2017/03/23/arts/global-art-market.html>
- [TOO] D. Adams. 50 of the Most Brutal Simon Cowell Quotes Ever. *The Odyssey Online*. 2017. <https://www.theodysseyonline.com/50-the-most-brutal-simon-cowell-quotes-ever>
- [MMM] K. Richman-Abdou. Art History: What is Contemporary Art? *My Modern Met*. 2017. <https://mymodernmet.com/what-is-contemporary-art-definition/>
- [ANN] S. Cascone. AI-Generated Art Now Looks More Convincingly Human Than Work at Art Basel, Study Says *ArtNet News*. 2017. <https://news.artnet.com/art-world/rutgers-artificial-intelligence-art-1019066>



CONFIDENTIAL COMMITTEE MATERIALS

SIGBOVIK 2018 Paper Review

Paper 14: Automating Art Snobbiness

Simon Cowell

Rating: Thank you. No.

Confidence: There is as much chance of this paper getting accepted as me flying to the moon tomorrow morning for breakfast. It's never going to happen.

If you had lived 2,000 years ago and done science like that, I think they would have stoned you. The only decent thing about that paper was the end.

Toward a historically faithful performance of the piano works of Antonín Qweřtý

William Gunther

Google, Inc.

wgunther@google.com

Brian Kell

Google, Inc.

bkell@google.com

SIGBOVIK '18

Carnegie Mellon University

April –2, 2018

Concrete

The great Czech composer Antonín Dvořák (1841–1904) wrote many pieces for the piano, including the famous *Humoresque No. 7 in G-flat Major* [2]. Unfortunately, typical performances of these works today sound nothing like what the composer intended because most modern pianos are configured with a different keyboard layout. Through painstaking historical research, we have reconstructed the original Dvořák piano keyboard layout. We have applied this discovery by transposing the *Humoresque* so that it is playable on a modern piano, enabling the first historically faithful performance of this piece in over a century.

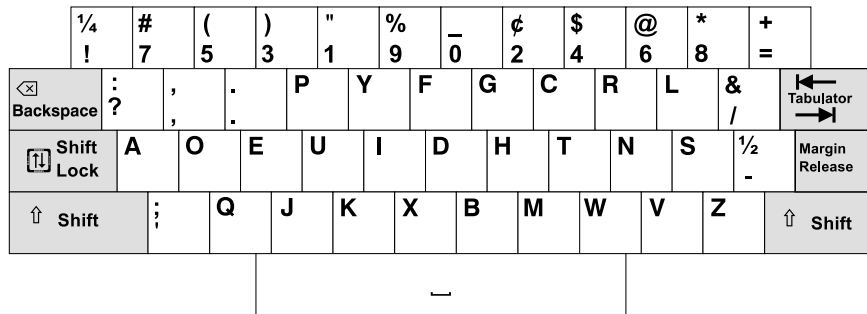


Figure 1: A Dvorak keyboard with the original or “classic” layout [3]. There are several variants of the Dvorak layout, but Dvořák was a classical composer, so this is almost certainly the one he used. Furthermore, this layout has 44 white keys (not counting the spacebar, which is clearly used only for rests). That is exactly half of the number of keys on a piano. Thus we may confidently conclude that the left half of Dvořák’s piano layout was just these 44 keys, while the right half was the same keys again with the Shift key held down.

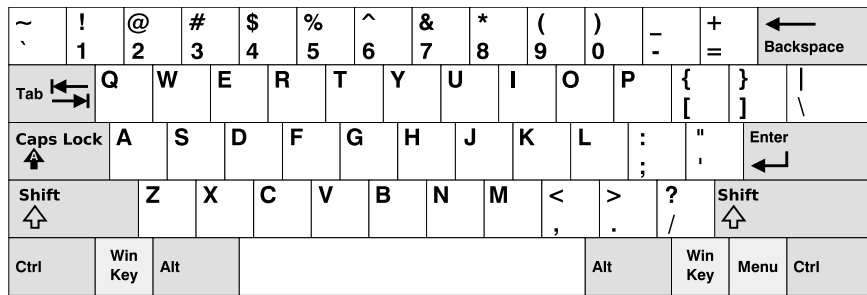


Figure 2: A modern QWERTY keyboard with the United States layout [4]. This layout has 47 white keys (not counting the spacebar), but obviously three of them are useless: nobody really needs the characters ‘~]}| [1]. This leaves 44 keys in the same positions as the keys of the Dvorak keyboard, which can then be mapped to the piano in the same way.

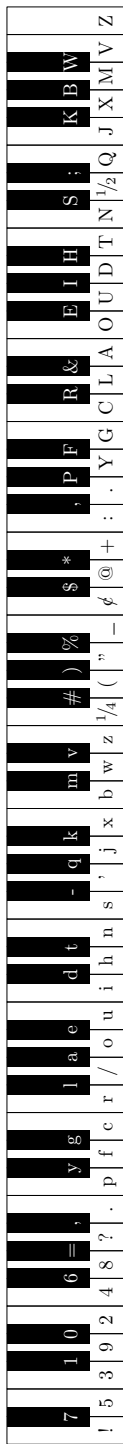


Figure 3: The reconstructed Dvořák keyboard layout. Although it looks strange to modern eyes, this keyboard would have looked familiar to Antonín Dvořák in 1894 and is unquestionably the layout for which he composed the *Humoresques* and his other piano works.

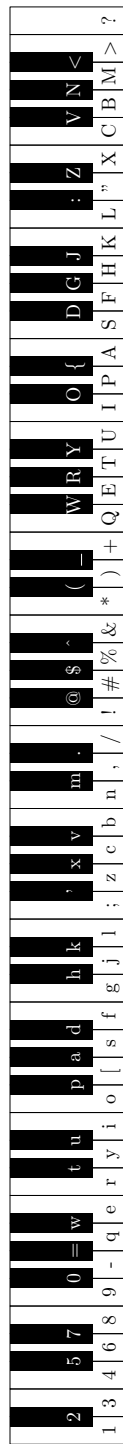


Figure 4: The familiar Qwertý keyboard layout, used by most modern pianos. The mapping from keys of the Dvořák keyboard to those of the Qwertý keyboard is in most cases obvious. The main difficulty lies in the keys $1/4$, ϕ , and $1/2$ on the Dvořák keyboard, which do not appear on the Qwertý keyboard. While a modern Qwertý piano cannot reproduce the note $1/4$ exactly, we can come close by playing 1, /, and 4 simultaneously. Likewise, we can approximate ϕ and $1/2$ with the chords {c, /} and {1, /, 2}, respectively.

Humoresque No. 7 in G-flat Major

Antonín Dvořák

Poco Lento e grazioso.

leggiere

* senza

Red.

Red.

3

5

4

p

dimin.

Red.

Red.

6

pp

Red.

Red.

Red.

9

f

dimin.

Red.

Red.

Red.

12

p

Red.

Red.

Red.

Red.

95

15 *ritard.* *in tempo*
fz *dimin.* *pp*
Red.

18 *Red.* *Red.*

21 *cresc.* *Red.* *Red.*

23 *ritard.* *Red.* *Red.* *Red.* *Red.* *Red.*

26 *mf* *dim.* *f*
96

30

fz *dim.*

Red. Red.

This system contains measures 30 through 33. The music is in a key with three sharps (F#, C#, G#) and a 2/4 time signature. The right hand features a melodic line with slurs and a dynamic marking of *fz* (forzando) at the beginning, which then transitions to *dim.* (diminuendo). The left hand provides a bass line with chords and single notes. There are two 'Red.' markings below the bass line in measures 32 and 33.

34

Red. Red. Red. Red. Red.

This system contains measures 34 through 36. The right hand continues with a melodic line, and the left hand has a bass line with chords. There are five 'Red.' markings below the bass line in measures 34, 35, 36, 35, and 34 respectively.

37

Red. Red. Red. Red. Red. Red.

This system contains measures 37 through 40. The right hand has a more complex texture with chords and slurs. The left hand continues with a bass line. There are six 'Red.' markings below the bass line in measures 37, 38, 39, 40, 39, and 38 respectively.

40

dim. *pp*

Red. Red. Red. Red.

This system contains measures 40 through 42. The key signature changes to two flats (Bb, Eb) starting in measure 41. The right hand has a melodic line with slurs and a dynamic marking of *dim.* in measure 40, and *pp* (pianissimo) in measure 41. The left hand has a bass line with chords and asterisks in measures 41 and 42. There are four 'Red.' markings below the bass line in measures 40, 41, 42, and 41 respectively.

43

Red. Red. Red. Red.

This system contains measures 43 through 45. The key signature remains two flats. The right hand has a melodic line with slurs and a dynamic marking of *pp* in measure 43. The left hand has a bass line with chords and asterisks in measures 43 and 45. There are four 'Red.' markings below the bass line in measures 43, 44, 45, and 44 respectively. A page number '97' is located at the bottom center of this system.

46

ritard.

Red. * *Red.* *Red.* *Red.*

48

in tempo

Red. *f*

51

dimin. *p*

Red. *Red.*

54

dim. *ritard.* *p dim.* *pp*

Red. *Red.* *Red.*

9

Musical score for measures 9-11. The piece is in a key with three flats (B-flat major or D-flat minor) and a 3/4 time signature. Measure 9 features a piano introduction in the bass line. Measure 10 has a forte (*f*) dynamic. Measure 11 is marked *dimin.* (diminuendo). The right hand has a melodic line with slurs and ties, while the left hand provides harmonic support with chords and moving lines.

12

Musical score for measures 12-14. Measure 12 starts with a piano (*p*) dynamic. Measure 13 has a piano introduction in the bass line. Measure 14 has a piano introduction in the bass line. The right hand continues with a melodic line, and the left hand provides harmonic support.

15

Musical score for measures 15-17. Measure 15 has a piano introduction in the bass line. Measure 16 is marked *ritard.* (ritardando) and *fz* (forzando). Measure 17 is marked *in tempo* and *pp* (pianissimo). The right hand has a melodic line with slurs and ties, and the left hand provides harmonic support.

18

Red. * Red. *

21

Red. * Red. *

23

Red. Red. Red. Red. Red.

26

mf

dim.

f

This system contains measures 26 through 29. The music is in a key with three sharps (F#, C#, G#) and a common time signature. The right hand features a melodic line with slurs and ties, while the left hand provides a harmonic accompaniment. Dynamic markings include *mf* at the start, *dim.* in measure 28, and *f* in measure 29. A hairpin symbol is visible in measure 28.

30

fz

dim.

Red. Red.

This system contains measures 30 through 33. The right hand continues with a melodic line, and the left hand has a more active accompaniment. Dynamic markings include *fz* in measure 30 and *dim.* in measure 32. The word "Red." appears below the bass staff in measures 32 and 33.

34

Red. Red. Red. Red. Red.

This system contains measures 34 through 37. The right hand has a melodic line with slurs, and the left hand has a steady accompaniment. The word "Red." is written below the bass staff in measures 34, 35, 36, 37, and 38.

37

Red. Red. Red. Red. Red. Red.

This system contains measures 37 through 40. The music is in a key with three sharps (F#, C#, G#) and a common time signature. The right hand features complex, multi-measure chords with accents and slurs. The left hand has a steady bass line with repeated notes and rests. The word "Red." is written below the bass line in measures 37, 38, 39, and 40.

40

dim. pp Red. Red. Red.

This system contains measures 40 through 43. The key signature changes to two sharps (F#, C#) in measure 40. The right hand has a melodic line with slurs and accents, marked with "dim." and "pp". The left hand continues with a bass line, marked with "Red." and asterisks. The word "Red." appears below the bass line in measures 40, 41, 42, and 43.

43

Red. Red. Red. Red.

This system contains measures 43 through 46. The key signature changes to two flats (Bb, Eb) in measure 43. The right hand has a melodic line with slurs and accents. The left hand has a bass line with repeated notes and rests, marked with "Red." and asterisks. The word "Red." appears below the bass line in measures 43, 44, 45, and 46.

46

ritard. Red. Red. Red. Red.

This system contains measures 46 through 50. The key signature remains two flats (Bb, Eb). The right hand has a melodic line with slurs and accents, marked with "ritard.". The left hand has a bass line with repeated notes and rests, marked with "Red." and asterisks. The word "Red." appears below the bass line in measures 46, 47, 48, 49, and 50.

48 *in tempo*

f

Red.

This system contains measures 48, 49, and 50. Measure 48 begins with a piano reduction (*Red.*) in the bass clef. The right hand features a melodic line with slurs and ties. Measure 49 continues the melodic development. Measure 50 concludes with a fortissimo (*f*) dynamic marking.

51

dimin.

p

Red.

Red.

This system contains measures 51, 52, and 53. Measure 51 includes a *dimin.* (diminuendo) instruction. Measure 52 features a piano (*p*) dynamic. Measure 53 ends with a piano reduction (*Red.*) in the bass clef.

54

dim.

ritard.

p dim.

pp

Red.

Red.

Red.

This system contains measures 54, 55, and 56. Measure 54 includes a *dim.* instruction. Measure 55 features a *ritard.* (ritardando) instruction. Measure 56 concludes with a pianissimo (*pp*) dynamic and a piano reduction (*Red.*) in the bass clef.

References

- [1] Bringhurst, Robert. *The Elements of Typographic Style*, version 3.1. Hartley & Marks, 2005. For example, Bringhurst dismisses the tilde key: “In the eyes of ISO and Unicode, the swung dash found on computer keyboards is an *ascii tilde*—a character . . . meaningless to typographers.” He describes the backslash as “an unsolicited gift of the computer keyboard” with “no accepted function in typography.” And of the pipe character, he writes, “Despite . . . its presence on the standard ASCII keyboard, the pipe has no function in typography. This is another key, and another slot in the font, that begs to be reassigned to something typographically useful.”
- [2] Dvořák, Antonín. *Humoresque No. 7 in G-flat Major*, Op. 101, S. 123. N. Simrock, London, 1894. Reprinted in *Humoresques & Other Works for Solo Piano*, Dover Publications, 1994.
- [3] Optikos at English Wikipedia. *File:KB DSKtypewriter.svg*. February 10, 2010. https://commons.wikimedia.org/wiki/File:KB_DSKtypewriter.svg. Licensed under the Creative Commons Attribution-Share Alike 3.0 Unported license and the GNU Free Documentation License.
- [4] Wikimedia Commons contributors (Denelson83, Bodigami, Bencherlite, and Yes0song). *File:KB United States.svg*. December 27, 2010. <https://commons.wikimedia.org/wiki/File:KB.United.States.svg>. Licensed under the GNU Free Documentation License and the Creative Commons Attribution-Share Alike 3.0 Unported license.

Thanks to William Lovas for his piano performance during our presentation at the SIGBOVIK conference.



CONFIDENTIAL COMMITTEE MATERIALS

SIGBOVIK 2018 Paper Review

Paper 21: Toward a historically faithful performance of the piano works of Antonín Qweřtý

Dr. Tom Murphy VII Ph.D.

Rating: 100 BPM

Confidence: Fortissimo

review.mid

the program committee
SIGBOVIK 2018



WordTeX

A WYSIPCTWOTCG¹ Typesetting Tool

Tom Wildenhain

Abstract

[WordTeX](#) is a plugin for Microsoft Word that attempts the impossible: creating documents that appear to be written in \LaTeX while irritating people who like \LaTeX . It is both stupidly impractical and surprisingly useful, offering an editing experience that is initially more enjoyable than \LaTeX and Word but is asymptotically more complicated than either. In this paper, I will explain the results of my [WordTeX](#) research. I will occasionally include content that has no relevance to the paper and is simply used to showcase how [WordTeX](#) renders certain elements. $(x + 1)^2 = x^2 + 2x + 1$.

1 Introduction

\LaTeX is a popular typesetting tool for creating complicated, consistently-formatted documents. Many students and scientists use \LaTeX because the finished documents have a clean, professional look that is hard to achieve in standard word processors.^[citation needed] However, these high-quality results come at the cost of a steep learning curve, potentially tedious editing experience, and sudden anxiety when homework assignments won't compile before a deadline.

Microsoft Word is a WYSIWYG editor, enabling users to have confidence in the appearance of their documents throughout the editing process. In fact, (real) research shows that Word novices are more productive and

¹ What You See Is Pretty Close To What Other Tools Can Get

make fewer errors than \LaTeX experts when creating certain types of documents.²

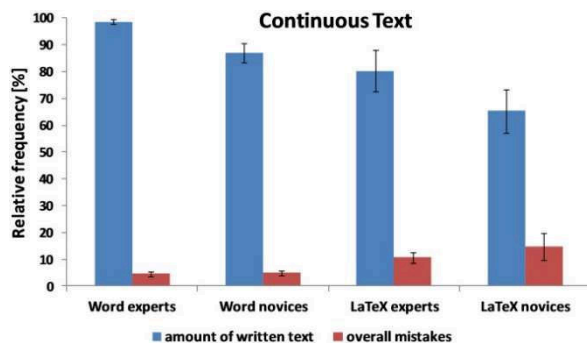


Figure 1: Word users are more productive than \LaTeX users for some document types. (Knauff & Nejasmic 2014)

However, when authoring more complicated documents, Word users spend much of their time clicking through menus and fiddling with formatting, leading to inconsistent document structure.

Word \TeX aims to combine the strengths of Word and \LaTeX , creating high-quality documents in real time, while annoying both Word and \LaTeX fans. It is particularly useful for typesetting homework assignments, although students should be warned that professors may warn students that “using \LaTeX is an important skill that will help you in the long term.”

2 The **Word \TeX** Template

The main component of **Word \TeX** is the WordTeX.dot template file. The template includes a set of styles that closely approximate the appearance of \LaTeX . The template is designed for Microsoft Office Word 2016 for Windows (or Office 365 ProPlus) and might not work correctly in other versions of Word.³

² Knauff, M., & Nejasmic, J. (2014). An Efficiency Comparison of Document Preparation Systems Used in Academic Research and Development.

³ I am not yet sponsored by Microsoft.

2.1 Fonts

Word \TeX users should install the LM Roman fonts and Latin Modern Math font (published by GUST) before using the template.

2.2 Styles

Word \TeX uses styles to maintain a consistent look throughout the document. Styles are provided for the title, subtitle, inline code, headings, and other common \LaTeX elements (Figure 2).

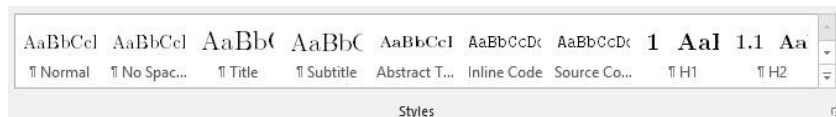


Figure 2: The style gallery

A user can apply a style by selecting it from the style bar or by typing `ctrl+shift+S` and then the name of the style (the latter is much faster). Typing `ctrl+shift+Z` clears character styles (which are applied within a line). Switching to the Normal style clears paragraph styles. Heading styles are automatically numbered, and styles throughout the document are updated if a style is modified.

2.3 Math

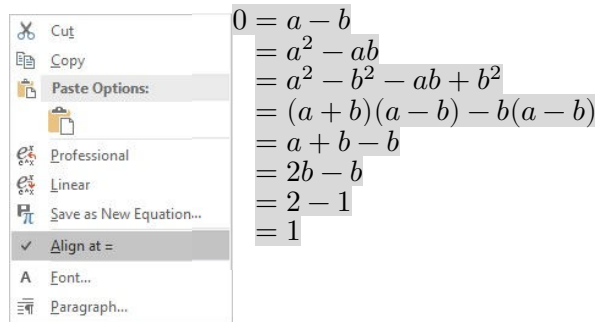
Math in **Word** \TeX uses the Latin Modern Math font. To enter or exit math mode a user can type `alt+=`. Latex commands like `\sum` and `\subsetq` are supported and render in real time. Word uses parentheses for grouping instead of curly braces. For example, `\sum_{i=1}^{10} i=55` renders as:

$$\sum_{i=1}^{10} i = 55$$

2.3.1 Alignment

A set of aligned equations can be created by selecting “align at =” from the equation context menu.

Let $a = b \neq 0$.


$$\begin{aligned} 0 &= a - b \\ &= a^2 - ab \\ &= a^2 - b^2 - ab + b^2 \\ &= (a + b)(a - b) - b(a - b) \\ &= a + b - b \\ &= 2b - b \\ &= 2 - 1 \\ &= 1 \end{aligned}$$

2.3.2 Functions

Functions like $\sin(x)$ and $\log_2(x)$ are automatically written without italics if they appear in the recognized function list. The equation options dialog (accessible from the conversions section) can be used to add more functions to the list.

2.3.3 Blackboard Bold

Blackboard bold letters like \mathbb{R} and \mathbb{N} can be inserted by typing `\doubleR` or `\doubleN`. They unfortunately look different from the L^AT_EX Blackboard Bold font, since they use glyphs from the Latin Modern Math font. Future researchers might be able to edit the font to include the `\mathbb` glyphs. The commands for these symbols can be shortened (see the section on macros).

2.4 Proofs

Proofs are started with the word “proof” set in the proof character style. To end a proof, a user can type `\tab` and then `\qed` to insert a \square symbol.

Proof. This is a proof. It is written in [WordT_EX](#). \therefore [WordT_EX](#) can make proofs. \square

2.5 Code

```
def print_code_instructions():
    assert includes_styles(inline and block_code)
    # Syntax highlighting support is planned and will
    # hopefully be released soon.
```

2.6 Macros

L^AT_EX includes a powerful macro system allows users to define custom commands. Word’s AutoCorrect is somewhat similar, but only performs basic text replacement. For example, you can add a math AutoCorrect entry that replaces `\R` with \mathbb{R} , but AutoCorrect entries can’t take arguments.

Despite these limitations, AutoCorrect is in fact exactly as powerful as L^AT_EX macros, as they are both Turing Complete.⁴ We can easily simulate a Turing Machine using AutoCorrect entries by representing the state as a string that “reads” characters by adding different entries for every combination of adjacent characters.

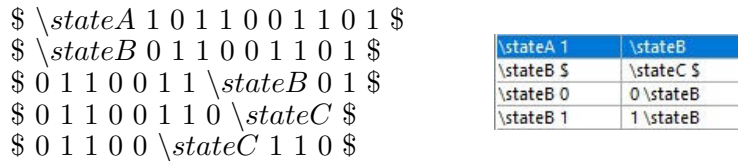


Figure 3: An example computation trace. `$` marks the ends of the input.

Unfortunately, AutoCorrect macros only evaluate once from left to right, so the Turing Machine will stop running if it ever moves to the left. To continue evaluation, a user can repeatedly press the Convert button in the Equation Tools tab. Be warned: if the Turing Machine moves to the right without halting, evaluation will not stop, and Word will freeze. I do not know if Microsoft is aware of this issue.

2.7 Printing

While Word has a built-in export to PDF option, it unfortunately does not embed otf fonts (like the Latin Modern fonts). The best option is to print to the Microsoft Print to PDF printer, which will embed the fonts.

⁴ <https://www.sharelatex.com/blog/2012/04/24/latex-is-more-powerful-than-you-think.html>

3 Conversions

While `WordTeX` is superior to `LaTeX` in many ways, sometimes `LaTeX` source is required for a conference or assignment.⁵ Thanks to Pandoc, `WordTeX` files can be converted to `LaTeX` source code.⁶ Mathematical expressions ($\int_0^4 x dx = 8$), `inline code`, and most *formatting* is converted. In fact, here's the source for this paragraph:

```
\hypertarget{conversions}{%
\section{Conversions}\label{conversions}}

While WordTeX is superior to LaTeX in many ways, sometimes LaTeX source
is required for a conference or assignment.\footnote{Fortunately,
  SIGBOVIK does not have such ridiculous restrictions.} Thanks to
Pandoc, WordTeX files can be converted to LaTeX source code.
\footnote{\url{https://pandoc.org/}}
Mathematical expressions ( $\int_0^4 x dx = 8$ ),
\texttt{inline\ code}, and most \emph{formatting} is converted.
In fact, here's the source for this paragraph:
```

The `WordTeX` plugin adds Copy as LaTeX and Paste From LaTeX buttons to Word. They use Pandoc to convert between formats in real time. The conversion isn't perfect, but is fairly close.

4 Similarity to LaTeX

4.1 Experiment

I conducted a double-blind randomized study to determine whether documents typeset using `LaTeX` and `WordTeX` are distinguishable (Fig. 4). The test subject was blindfolded, and the experimenter (also blindfolded) told her to write L or W on each paper she believed to be a `LaTeX` or `WordTeX` document, respectively. Data was collected until the results supported the hypothesis that `LaTeX` and `WordTeX` are indistinguishable.

⁵ Fortunately, SIGBOVIK does not have such ridiculous restrictions.

⁶ <https://pandoc.org/>

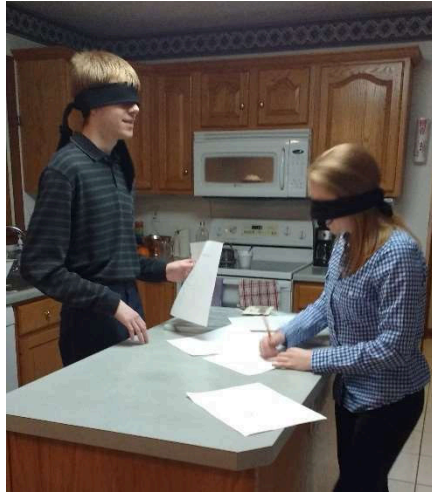


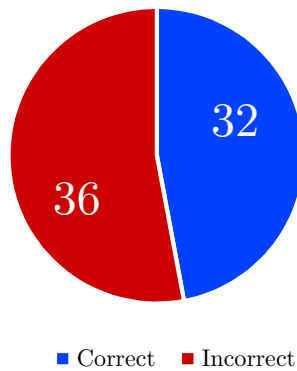
Figure 4: Experimental setup

4.2 Results

The papers were misidentified most of the time.

	L ^A T _E X	WordT _E X
Identified as L ^A T _E X	16	18
Identified as WordT _E X	18	16

It is clear from the following chart that the red side is bigger than the blue side ($p < 0.5$).



4.3 Conclusions

Running a χ^2 test, the p -value is 0.628, which is not significant. Therefore, documents typeset in \LaTeX do not appear to be significantly different from those typeset in [Word \$\text{\TeX}\$](#) .

5 Summary

[Word \$\text{\TeX}\$](#) is a typesetting system that supports the basic functionality of \LaTeX while utilizing the editing convenience of Word. Word's Turing-complete macros ensure that [Word \$\text{\TeX}\$](#) is just as powerful as \LaTeX (for questionable definitions of “powerful”). The [Word \$\text{\TeX}\$](#) plugin allows for easy conversion between [Word \$\text{\TeX}\$](#) and \LaTeX . Experimental results suggest that [Word \$\text{\TeX}\$](#) and \LaTeX documents are indistinguishable. In light of these results, I encourage all scientists, students, and professors to abandon \LaTeX immediately and use [Word \$\text{\TeX}\$](#) for future work.



CONFIDENTIAL COMMITTEE MATERIALS

SIGBOVIK 2018 Paper Review

Paper 12: Word $\text{T}_{\text{E}}\text{X}$
A WYSIPCTWOTCG Typesetting Tool

Stephan Jenkins and Kevin Cadogan

Rating: Impressive but concerning

Confidence: 2/3

The paper presents an impressive system for typesetting $\text{T}_{\text{E}}\text{X}$ -style documents in Microsoft Word. However, we are concerned with the validity of the double-blind study. While reading the paper, we were able to see the labels on the table presenting the results. A truly unbiased study would be triple-blind, hiding the labels not just from the subject and experimenter but also the reader.

Donald Knuth

Rating: Badness 10000

Confidence: Overfull

I could just use Calibri and make my $\text{T}_{\text{E}}\text{X}$ look like Word if I weren't typing this on a Mac.



SIGBOVIK 2018

(Continued) Message from the Organizing Committee

“Dude, don’t try to fool me,” whispers CoBot. “I know a robot when I see one. Come on, I’ll show you to SIGBOVIK.”

goto PAGE_208;



SIGBOVIK 2018

(Continued) Message from the Organizing Committee

You emphatically shrug, along with most of the other humans, none of whom seem to know what the Charlie Brown dance move is. An interrupt from your starboard camera informs you of an approaching obstacle. Upon closer examination, you find that the obstacle is the cluster of humans who were watching you dance. They must be Serious Researchers trying to reprogram you! You try to roll away, but it's too late—one of the humans claps his hand around you.

“Hey, we saw you totally killing the Cha Cha Slide out there.”

“Yeah, you really nailed the Charlie Brown shrug.”

“How about we ditch this party and do some real dancing?”

Determining that dancing is unlikely to lead to reprogramming, you accept the humans' invitation.

You roll with the humans to an arcade and gather with them around a shiny new ITGAKADDR++ machine. “This game is very simple: just step on the pads in time to the music as indicated by the arrows on the screen,” one of the humans explains. “It should come naturally to a Cha Cha Slide expert like you.” They give you a recent paper on dance games [3] to get you up to tempo.



```
switch (choose_dear_reader()) {  
case FANTASTIC:  
    Press the down, left, right, and down pads in that order.  
    goto PAGE_87;  
case DECENT:  
    Press the up, down, right, and right pads in that order.  
    goto PAGE_50;  
case MISS:  
    Don't press any pads.  
    goto PAGE_28;  
}
```


Systems

Wheel

17 mallocd: designing a garbage-free nosql data store

igor

Keywords: state-of-the-art, in-memory, garbage-free, cloud-native, nosql, bare metal, containerless, micro-service, data store

18 The fluint8 software integer library

Jim McCann and Tom Murphy VII

Keywords: uint8_t, float, binary32, ieee754

19 A survey of hardware multithreading

Sol Boucher

Keywords: bend, the, kneedle

mallocd: designing a garbage-free nosql data store

igor

2018-04-01

abstract

at bigcorp (tm) we have a lot of data. many jiggabytes worth of data. in order to get the data fast, we need to put it in memory. that is because memory is faster than other forms of storage. in this paper, we introduce a new design for a state-of-the-art, in-memory, garbage-free, cloud-native, nosql, bare metal, containerless, micro-service, data store.

the shift key on my keyboard is broken.

1. garbage

certain “professional” models of a particular brand of computing device have been said, by some, to be aesthetically similar to trash cans. this comes as no surprise, indeed, if you look at many “desktop” screens at the office, you will see a similar pattern: the trash can icon is empty, and the entire desktop instead has been littered with garbage.

this is also quite similar to how modern software operates. as the old saying goes: when you leak the trash, you seek and thrash. garbage collectors have been at our disposal for many years now.

before the sunset, the coffee engineers at some microsystems corporation came across a paper about symbolic expressions and their computation by coffee machine. after a bit of small talk, they got to work, and eventually produced a device capable of briefly stopping time. the garbage collector.

this stopping of time occurs when too much garbage is produced. and when time stops in our data store, we cannot get the data very fast.

2. go

all cool new technology is written in go. for this very reason, we chose to use go as the foundation for our new data store.

all cool new technology written in go has a `.io` top-level domain. we tried to get approval from corporate to purchase `mallocd.io`, but the request was denied. as a result, we are currently seeking funding. if you are an investor with \$30 bucks to spare, please get in touch.

go is as fast as c, because it was created by the fresh prince of bell labs. except when the garbage collector runs. go employs a “mark-and-weep” collector. this is sometimes also referred to as a “tracing” collector, named after the traces left by the tears rolling down the cheeks of those who are waiting.

n.b. this is also why distributed tracing systems are used in multi-tear architectures.

fun fact: there was a mistake in the original c programming language that ended up costing a billion dollars. but because the mistake was so iconic, they ended up including it in go as well! who knew?

3. allocation

in order to create garbage, things need to be put somewhere. in the context of programming, this process is called memory allocation.

the good old `stdlib.h` defines the following function

signatures:

```
void *malloc(size_t size);
void free(void *ptr);
void *calloc(size_t nmemb, size_t size);
void *realloc(void *ptr, size_t size);
```

`malloc(3)` is used for creating garbage, and `free(3)` is used for collecting it. the reason you've never heard of these functions is because the garbage collector does the work for you.

freeing memory is important, because otherwise the kernel will stop the database process and you will get paged at an inconvenient time. as the old saying goes: when you free your memory, you also free your mind.

4. democratizing malloc

the `malloc(3)` machinery for direct memory allocation allows dealing with memory directly. most contemporary programming languages explicitly deny their users the opportunity to mess with arbitrary memory locations.

what if we had a mechanism that would allow *anyone* to use `malloc(3)`?

this is how we got the idea of `mallocd`. `mallocd` is a stateful micro-service written in go that provides direct memory access for high-performance data storage and retrieval.

the “d” in `mallocd` refers to “daemon”, as this service allows its users to harness demonic powers.

the `mallocd` service exposes a udp socket, allowing *any* other process to allocate, access, modify, and free, memory.

5. client

clients for `mallocd` can be written in any language. you could use a command-line client that lets you allocate memory from the comfort of your couch.

here is an example of what that would look like:

```
$ mallocd-client malloc 5
842350568512
$ mallocd-client write 842350568512 5 hello
$ mallocd-client read 842350568512 4
hell
$ mallocd-client free 842350568512
```

don't forget to free your pointers!

6. protocol

`mallocd` uses a binary protocol. mainly for performance reasons, but also so that we could create one of those cool diagrams you see in rfc's.

the protocol exposes 4 methods: `malloc`, `free`, `read`, `write`.

6.1 malloc

the `malloc` request (0x00) is used to allocate memory of `len` bytes. it returns a 64-bit pointer to that memory `ptr`.

request:

```
  0 1 2 3 4 5 6 7
+-----+
|           |
|    0x00   |
|           |
+-----+
|           |
|    len    |
|           |
+-----+
```

reply:

```
  0 1 2 3 4 5 6 7
+-----+
|           |
|    ptr    |
|           |
+-----+
```

these diagrams are not what they seem. what you thought were bits are in fact bytes. we figured, since we're sending huge 64-bit addresses around, we might as well make all fields 64 bits wide. it still looks really cool though.

6.2 free

the `free` request (0x01) is used to free the memory allocated at the memory location pointed at by the pointer `ptr`.

request:

```

 0 1 2 3 4 5 6 7
+-----+
|      0x01      |
+-----+
|      ptr       |
+-----+

```

this request has no reply. we simply assume the udp message was received and the memory was freed successfully.

this usually works.

6.3 read

the `read` request (0x02), not to be confused with the `read(2)` system call, allows reading an arbitrary chunk of memory of length `len` by de-referencing the pointer `ptr`.

request:

```

 0 1 2 3 4 5 6 7
+-----+
|      0x02      |
+-----+
|      ptr       |
+-----+
|      len       |
+-----+

```

reply:

```

 0 1 2 3 4 5 6 7
+-----+
|      + (len bytes) +
|
+-----+

```

this is mostly safe.

6.4 write

in order to write to any any address in the `mallocd` process, the `write` request (0x03) is used. just like `free`, this request does not have a reply.

```

 0 1 2 3 4 5 6 7
+-----+
|      0x03      |
+-----+
|      ptr       |
+-----+
|      len       |
+-----+
|      + (len bytes) +
|
+-----+

```

while all of our bits are very significant, we transmit bytes in big-endian order, which is the one that just makes sense.

7. garbage-free computing

with the boring stuff out of the way, let's talk about garbage-free computing.

the go runtime can be provided with a `GODEBUG` environment variable that can print out information about how much garbage a program is producing.

in order to compute without garbage, one must pre-allocate all structs and buffers, and then re-use them. this is precisely what `mallocd` does.

once allocated, no garbage collector will touch those buffers ever again.

unfortunately the go standard library's udp networking code is not entirely garbage-free. receiving udp datagrams results in the allocation of two structs: `syscall.SockaddrInet4` and `net.UDPAddr`.

we can work around this by making syscalls directly:

```

r1, _, e := syscall.Syscall6(
    syscall.SYS_RECVFROM,
    fd,
    uintptr(unsafe.Pointer(&req[0])),
    uintptr(len(req)),
    0,
    uintptr(unsafe.Pointer(&addr)),
    uintptr(unsafe.Pointer(&addrSize))
)

```

now the only garbage produced in `mallocd` is memory allocated by `malloc` requests.

8. manually managing memory

to allocate memory, the `reflect` package can be used as usual:

```

t := reflect.ArrayOf(
    int(len),
    reflect.TypeOf(byte(0))
)
ptr := reflect.New(t).Pointer()

```

`ptr` can now be handed out for anyone to use freely.

however, that allocated memory is now at risk of being collected. in order to prevent that from happening, we keep a reference to it in a shared map:

```

p := unsafe.Pointer(uintptr(ptr))
refs[p] = nil

```

later, the memory can be freed by removing its reference from the map:

```

p := unsafe.Pointer(uintptr(ptr))
delete(refs, p)

```

this is a subtle way of instructing the garbage collector to run `free(3)` on that pointer and reclaim the memory.

don't forget to free your pointers!

9. results

we will be rolling out `mallocd` to production at bigcorp (tm) next week.

10. conclusion

`mallocd` is a next-gen, best-in-class, garbage-free, in-memory, nosql datastore written in go.

it democratizes `malloc` by allowing *anyone* to mess with memory in *any* language.

since it's just memory, it's easy to implement your own data structures on top of `mallocd`.

don't forget to free your pointers!

11. future work

other data stores provide support for scripting via lua stored procedures. `mallocd` provides the ability to write to arbitrary memory, it may be possible to write into the stack segment in order to create and run user-defined functions.

in order to simplify the adoption of `mallocd`, we want to develop a posix-compliant drop-in replacement for `malloc(2)` that can be loaded via `LD_PRELOAD`. it could use a `SIGSEGV` signal handler to intercept memory accesses, or if that doesn't work, we might just make a kernel module.

references

here are a few pointers:

- 0xc42001a440
- 0xc4200cb9e0
- 0xc4201355c0

don't forget to free them when you're done.



SIGBOVIK 2018

(Continued) Message from the Organizing Committee

You press the 4 button, and the elevator starts moving. The elevator makes a stop on floor 3 for the few thousand people taking the elevator up one floor to class. “While we wait, perhaps you can help me with a mathematics problem that has been floating in the back of my head for a bit,” says CoBot. “What is $0.1 + 0.2$?” This is not your first rodeo: as a robot with an FPU in place of an ALU, you are well aware of the perils of rounding errors. Fortunately, a recent paper describes a library for performing integer operations using only floating-point assembly instructions [6], and you had the library installed in your latest upgrade.

```
switch (choose_dear_reader()) {
case INT_COMES_NATURALLY:
    Use your new integer operations to reason  $0.1 + 0.2 = (1 + 2) * 0.1 = 0.3$ .
    goto PAGE_116;
case IEEE_754EVER:
    Use your classic FPU to report  $0.1 + 0.2 = 0.30000000000000004$ .
    goto PAGE_88;
}
```


The *fluint8* Software Integer Library

Jim McCann*
TCHOW llc

Tom Murphy VII†
tom7.org foundation

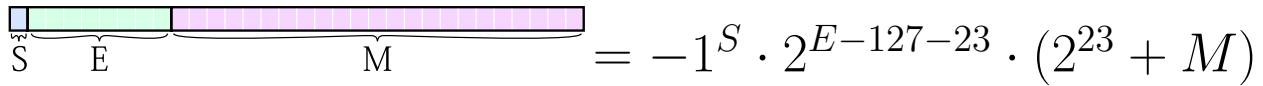


Figure 1: Our library performs unsigned integer operations using only arithmetic operations on IEEE754 floating point numbers stored in binary32 format (pictured).

Abstract

We present *fluint8*, a library for performing integer math, including basic arithmetic and bitwise logical operations, using only basic floating point operations.

CR Categories: (1.10100100011)₂ × 2¹¹ [Software]: Software Engineering—Coding Tools and Techniques

1 Introduction

There are a surfeit of libraries that exist to perform floating point operations on processors that only support integer math. This is unsurprising, as many such processors exist – from ancient 286’s to modern embedded microcontrollers. These libraries use many integer instructions to emulate the action of a floating point unit, providing correct and useful (if slow) results.

We present a small header-only C library to emulate integer operations – specifically 8-bit unsigned integer operations – using standard IEEE 754 single-precision (binary32) floating point math. Our presented operations have been designed to be succinct but also pleasantly puzzling.

As far as we are aware, no processor exists for which this library would be required. However, perhaps you should consider that a challenge.

2 Floating Point

An IEEE 754 single-precision floating point number (binary32 format) is stored as a sign bit, a 8-bit exponent, and a 23-bit mantissa (Figure 1). Except for special cases, the number represented by a floating point number with sign S , exponent E , and mantissa M is ‡:

$$-1^S \cdot (1.M)_2 \cdot 2^{E-127}$$

*e-mail: ix@tchow.com

†e-mail: tom7@tom7.org

‡or at least this is what wikipedia says, so I’m going with that, and it seems to work out.

Particularly, notice that the leading “1” in the fraction is implicit in the representation (it is implied by a non-zero exponent[§]).

This means that the range of integers that can be represented (without loss of precision) is

$$[-2^{24}, 2^{24}] = [-16777216, 16777216]$$

which, conveniently, is far more than the [0, 255] range needed for storing 8-bit unsigned integers.

When floating point operations result in numbers that cannot be accurately represented, the results are rounded according to the current rounding mode. The default rounding mode assumed in this paper is *roundTiesToEven*. I would say that it does what you expect, but floating point numbers seldom manage that feat. Regardless, this rounding mode means that whenever a value is exactly halfway between two representable numbers, the number with a least-significant-bit of 0 is picked.

Rounding and precision loss leads to this fun fact:

$$\begin{aligned} 16777216.0f + 1.0f - 1.0f &== 16777215.0f \\ 16777216.0f - 1.0f + 1.0f &== 16777216.0f \end{aligned}$$

(Hot take: floating-point operations are non-commutative.)

3 The *fluint8* Library

The *fluint8* library provides all of the mathematical and logical operations one expects on 8-bit integers, using only floating point addition, subtraction, multiplication, and division – other than a loop with fixed bounds which could be unrolled by the compiler, no conditionals or function calls are required.

Full source code for the library (and this paper) are available at <https://github.com/ixchow/fluint8>.

In this section we go through the library operation by operation, explaining how each function works.

[§]An all-zero exponent is used for special numbers like zero, but we’re not going to go into that. Wait, we just did.

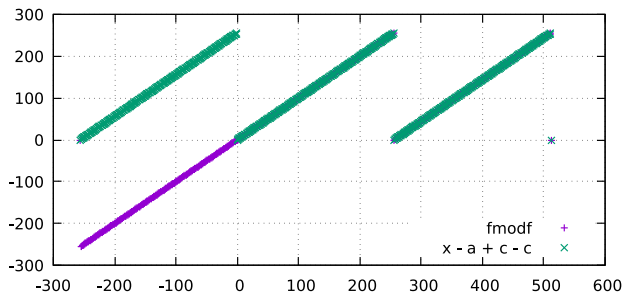


Figure 2: Comparing $\text{fmodf}(x, 256.0f)$ to the expression $x - 127.5f + 3221225472.0f - 3221225472.0f$ over the range $[-256.0f, 512.0f]$. Notice that the expression is positive for negative numbers, making it more useful for simulating integer rollover. Plot created using gnuplot.

3.1 Storage Format

Our library represents unsigned 8-bit integers as their equivalent floating point values. In other words, the value `uint_t(127)` is represented as `127.0f`. This straightforward equivalence is convenient when writing basic mathematical functions.

In order to support, e.g., reading data from files, our library includes functions that convert between floating point numbers and bit-patterns of their equivalent 8-bit unsigned representation.

```
void fu8_to_bits(float a, void *out) {
    a += 8388608.0f;
    memcpy(out, &a, (size_t)(1.0f));
}
```

The function `fu8_to_bits` adds a large enough number to `a` that its mantissa's least-significant bit now represents 1. Essentially, the code is shoving the integer information stored in `a` to the least-significant-byte of the representation, and then copying[¶] it out to the destination.

The same trick works when setting a floating point number from an integer bit pattern:

```
float fu8_from_bits(void const *from) {
    float a = 8388608.0f;
    memcpy(&a, from, (size_t)(1.0f));
    return a - 8388608.0f;
}
```

[¶] The astute reader will notice that we've taken care to avoid using an integer constant as a parameter to `memcpy`. Presumably on processors without integer support `size_t` must be a floating-point type. And, yes, we promised above not to use function calls, but it's hard to copy a byte without integer types.

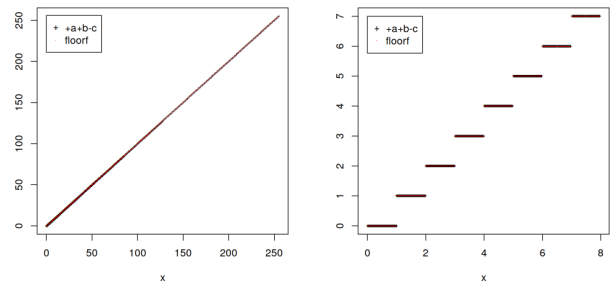


Figure 3: Comparing $\text{floorf}(x)$ to the expression $x + 0.50390625f + 8388608.0f - 8388609.0f$ over the all quotients $[0.0f, 255.0f] / [1.0f, 255.0f]$. The functions match at all plotted points. Plot created using R.

3.2 Arithmetic Functions

Our library implements `+`, `-`, `*`, `/`, `+`, and `-` by treating floating point numbers as real numbers; an approach that often works, but requires some post-processing to deal with rollover:

```
float fu8_add(float a, float b) {
    float x = a + b;
    x += x - 127.5f + 3221225472.0f -
    3221225472.0f;
    return x;
}
```

Here, the second line of the function computes $\text{fmodf}(x, 256.0f)$ by rounding `x` to the next-greater multiple of `256.0f` (rounding is forced by adding `3221225472.0f` to make the least-significant-digit of the number have value 256), then subtracting this rounded value. Don't believe us? Examine the convincing graph in Figure 2.

Most of the remaining math functions follow this “operate then wrap” paradigm:

```
float fu8_sub(float a, float b) {
    float x = a - b;
    x -= x - 127.5f + 3221225472.0f -
    3221225472.0f;
    return x;
}
float fu8_mul(float a, float b) {
    float x = a * b;
    x -= x - 127.5f + 3221225472.0f -
    3221225472.0f;
    return x;
}
float fu8_pos(float a) {
    return a;
}
```

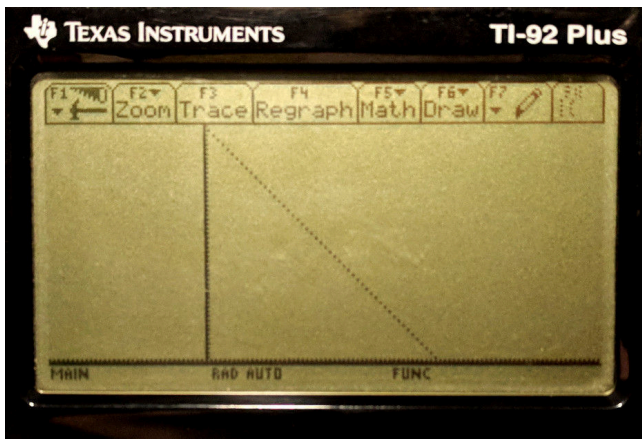


Figure 4: graph of $255.0f - x$ (the bitwise complement of x). Plot created using a TI-92 Plus graphing calculator.

```
float fu8_neg(float a) {
    return (a + 127.5f + 3221225472.0f -
            3221225472.0f) - a;
}
```

Our add-and-subtract modulus function always returns a positive number, which we take advantage of in the subtraction function. The negation function uses a similar trick to either subtract a from $256.0f$ if positive or from $0.0f$ otherwise.

```
float fu8_div(float a, float b) {
    float x = a / b;
    x = x + 0.50390625f + 8388608.0f -
        8388609.0f;
    return x;
}
```

The division function computes the floor of a value by rounding that value to the next-largest and subtracting one (Figure 3). In this particular instance, the constant required is small enough that the subtraction of one can be rolled into the subtraction of the large constant (we choose this over rounding down mostly for aesthetic reasons).

For the division and modulus operations, `fluint8` diverges from the normal behavior of integer instructions when the denominator is zero. A typical processor triggers a fault upon integer division by zero, but IEEE 754 instead returns one of the special values *Infinity*, *-Infinity* (or *NaN*) and continues calculating. After this point, `fluint8` may produce nonsense results. However, this is strictly compliant with the C and C++ standard, for which integer division by zero is formally undefined behavior.^{||}

^{||}“If the second operand of / or % is zero the behavior is undefined.” — C++03 5.6.4

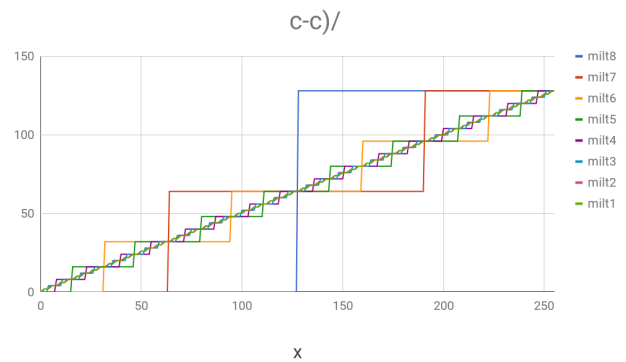


Figure 5: The value of $(a + 1.0f + c-c) / 2.0f$ for c ranging from $2147483648.0f$ (*milt8*) to $16777216.0f$ (*milt1*). Plot created in Google Sheets.

3.3 Bitwise Operations

Things really get interesting when we begin to look at bitwise operations, which aren’t standard operations on floating point numbers**.

Let’s begin with bitwise negation (`~`). This one is relatively easy to explain – an unsigned 8-bit integer plus its bitwise complement is always 255, which makes negation as easy as subtraction (Figure 4):

```
float fu8_not(float a) {
    return 255.0f - a;
}
```

Things get a bit more interesting when computing bitwise and (`&`):

```
float fu8_and(float a, float b) {
    float ax, bx, x = 0.0f;
    for (float c = 2147483648.0f;
         c != 8388608.0f; c *= 0.5f) {
        a -= ax = (a + 1.0f + c-c) / 2.0f;
        b -= bx = (b + 1.0f + c-c) / 2.0f;
        x = 0.5f * x + ax * bx;
    }
    return x;
}
```

Note that though this is presented as a loop, the loop has constant bounds and could be unrolled by a compiler into eight repetitions of the same code.

This code peels apart a and b bit-by-bit using a similar trick to the floating point modulus idea we explained earlier. In this case, however, we’ve formatted the code so it has a little waving guy in it, who we will call Milt:

$$c-c) /$$

**Though they seem well-defined; maybe a language-designer oversight

Though it looks like Milt is just hanging out, minding its own business, and not changing the value of the expression, Milt is in fact doing something surprisingly nonlinear (Figure 5).

So when Milt's eyes are $2147483648.0f$, it is extracting twice the value of the MSB of a , which in turn is stored in ax and subtracted from a . In this way, the code peels off each successive most-significant bit from a and b and accumulates their product into the final result.

This leaves only the mystery of why x is being divided by two each loop iteration. But this isn't a mystery at all. Consider computing $171 \& 226$. Notice that on the first iteration, the product $128.0f * 128.0f$ would be added to x ; the multiplications by a factor of $0.5f$ on each subsequent iteration simply – in aggregate – bring it to the correct result of $128.0f$.

a	b	ax	bx	x	c
171	226				
43	98	128	128	16384	2147483648
43	34	0	64	8192	1073741824
11	2	32	32	5120	536870912
11	2	0	0	2560	268435456
3	2	8	0	1280	134217728
3	2	0	0	640	67108864
1	0	2	2	324	33554432
0	0	1	0	162	16777216

4 Optimization

The previous routine computes a bitwise function a single bit at a time. While clean and logically motivated, it seems possible to improve bandwidth by processing multiple bits at once. As a proof-of-concept, the following routine computes the exclusive-or function (\wedge) for two `fluints` in the range 0-3.

```
float fu8_xor2bit(float a, float b) {
    return truncf(
        fmod(((a * -1.89269124e+30f) +
              (b * -1.09500709e+35f)) *
              -1.14474456e-18f,
              4.77664232f));
}
```

The routine works by computing a very noisy function that just happens to come close to the correct results for all 16 possible inputs. Don't believe us? Barbecue your eyes of Figure 6. This routine uses `fmod` and `truncf`, but the same loss-of-precision tricks from before can likely be used to avoid them. Four similar expressions can be composed to compute 8-bit exclusive-or, and/or it may be possible to find expressions that compute more bits at once.

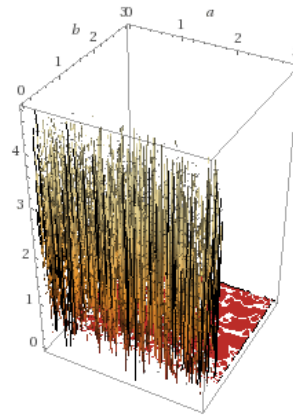


Figure 6: 3D graph of $((a \times -1.89269124 \times 10^{30} + (b \times -1.09500709 \times 10^{35})) \times -1.14474456 \times 10^{-18}) \bmod 4.77664232$ with a and b each ranging from 0–3.0. Plot created using Wolfram Alpha Computational Knowledge Engine.

5 Not Optimization

The library should not be used with compiler options such as `-ffast-math` (which may assume properties that do not hold of IEEE754, like commutativity). This often optimizes away the entire `fluint8` code, causing it to misbehave (whoa, not *that* fast, buddy).

6 Applications

While direct hardware applications of this technology are currently theoretical (Section 7), there is at least one compelling application for `fluint8` in everyday practice. Many pieces of software use primarily integer instructions, letting the floating point unit lie completely dormant for many nanoseconds at a time. As has been known for decades, while the integer registers and functional units are occupied, the otherwise idle floating point units can be used to perform useful tasks. Unfortunately, no useful tasks were known for floating point instructions. Now, we see that normal integer operations (such as cryptography) can be simulated with these instructions and registers. With compiler support, a separate thread of non-interfering floating point instructions could be emitted, and arbitrarily interleaved with the integer ones, scheduled only when the integer unit would likely stall for a data dependency. This technique is *ultrathreading*, since it is one step better than *hyperthreading*. For example, we hypothesize that during normal web browsing on a modern x86-64 processor, such code could mine as much as one Bitcoin per 107 trillion years, with no more than a 1% loss of efficiency.

7 Future Work

Design a processor for which this library is relevant.



CONFIDENTIAL COMMITTEE MATERIALS

SIGBOVIK 2018 Paper Review

Paper 18: The fluint8
Software Integer Library

Not a Number

Rating: Definitely a Number

Confidence: Could be a Number?

75035999728258881796215668736
184933246639179563008
283968873678046692484587716608
2.34475994176364110899157822132110595703125e-09



SIGBOVIK 2018

(Continued) Message from the Organizing Committee

You angle your wheel stalks $\pi/4$ radians apart to perform the Charlie Brown dance move when you hear a horrible ripping noise. Your pants suddenly feel much looser: you must have ripped them! Simulating embarrassment, you slowly roll to the side of the room and take out your sewing kit. The kit contains a single needle and single thread, but a recent paper on multithreading [4] suggests that it is advantageous to repair pants using multiple threads in parallel.

```
switch (choose_dear_reader()) {
case BY_A_SINGLE_THREAD:
    Attempt to repair your pants using a single thread.
    goto PAGE_154;
case WITH_HT_TECHNOLOGY:
    Find another thread-like object with which to repair your pants.
    goto PAGE_39;
}
```

A Survey of Hardware Multithreading

Sol Boucher
Carnegie Mellon University
sboucher@cs.cmu.edu

Abstract

I'm just going to assume you already read the title. We chose a needle as our hardware threading platform because the author had a rip in his pants. ~~More concretely~~, you'll have to read the paper for details; after all, this is just the abstract.

1 Introduction

This paper describes our first-hand experiences with needle threading in hardware. This paper is organized as follows: on second thought, why don't you just skim the section titles like a normal reader and work that out on your own?

2 Test Environment

We remind the reader that we were suffering from a pair of ripped pants at the time of writing. We hadn't expected the pair of pants in question to be ripped, and this misprediction led to a stall in our laundry pipeline. This resulted in a phenomenon we'll refer to as *cold legs*, so we began eyeing our bed as a potential location to continue the research. In the interest of full disclosure, there are a few admissions we should make about this test bed:

- It could have been better made.
- We briefly attempted to remedy this issue, but ultimately concluded that this is a hard problem.
- We conjecture that making the bed is actually an unimportant step, as our next step was to crawl in.
- For months now, we've been operating under the assumption that this conjecture is correct; however, it would be good to have rigorous proof.

3 Design

The core idea of multithreading is, of course, to improve efficiency by performing parallel work. To begin our investigation, we attached multiple threads to the same needle, as shown in Figure 1. While this sounds like a simple process, care must be taken to ensure that each individual thread is attached directly to the needle, as tying them end-to-end does not increase parallelism.



Figure 1: Multithreading

4 SMT

Before tying the threads to the needle, we discovered that they were asymmetrically threaded. We tried to balance them to achieve the desired symmetry, but this turned out to be imprecise when attempted by hand. We attempted to employ an SMT solver, but it seemed confused. Perhaps we should have paid more to hire one with better SAT scores, as we've been told that such solvers have quickly reported NP-complete ("No Problem, done") when faced with similar challenges.

5 Parallelism

To further motivate multithreading, we observe that the number of threads used in the test fabric is proportional to its resultant tensile strength. By Amdahl's law, increasing the parallelism decreases the time taken to achieve a given tensile strength. Needle-less to say, this trend would continue if our needle could accommodate more threads. As it is, we were only able to test at limited scale, but we posit that better hardware would allow us to quickly handle the Big Rip.

6 Coherency

In our experience, even the amount of parallelism achievable on our platform is difficult to manage. We struggled repeatedly as the, um, data became stuck in our KNOT gates. It wasn't until we dropped the needle, though, that the true extent of our catch coherency issues became apparent.

7 Future Work

We'd like to extend this work to test on platforms other than beds (e.g. haystacks). Although it seemed inappropriate for the repair in question, an investigation into threads that collect user input is sorely needed, not least because we have a bunch of spare buttons lying around. As more threads become involved, managing concurrency becomes increasingly critical; therefore, an investigation into the threading of eyebrows, locks, and other synchronization mechanisms is also in order.

8 Conclusion

As foreshadowed in Section 5, we suggest that future hardware include a larger loop thing so the needle can accommodate more threads. That said, the limited amount of parallelism we were able to achieve resulted in a pair of pants that allowed us to leave the test bed but not the house. We take the fact that we couldn't wear them in public as an indication that the problem is embarrassingly parallel.

Acknowledgements

We are grateful to the Lord for being our shepherd. Without Her assistance, the market economy would never have been able to capture enough sheep to supply the many threads required by our system.

References

"For the absence of a bibliography¹ I offer neither explanation nor apology."

¹or citation for this quotation



CONFIDENTIAL COMMITTEE MATERIALS

SIGBOVIK 2018 Paper Review

Paper EasierChair: A Survey of Hardware Multithreading

Jim McCann (CMU Textiles Lab)

Rating: 1.5

Confidence: High

While hardware multithreading is something that is relevant to both textiles construction and computer science, and I am a strong advocate for more textiles-cs crossover work appearing in important venues like SIGBOVIK, I believe this paper needs to be extended.

One could discuss, for example, a modern Jacquard loom, which uses a *warp*¹ of parallel threads that can be independently selected per-instruction.

Or, instead, take the odd case of a mechanical knitting machine, which, rather than operating on many threads with a single needle, operates on a single thread with many needles. Indeed, given the way knitting machines are constructed, many dependent operations can be in flight on a single thread at the same time, with the machine handling tension forwarding and global order resolution through careful construction.

Even in the consumer machine-sewing realm, it is common to see two-way multithreading: most machines use a “top thread” which passes through the needle and a “bottom thread” that comes from a smaller spool called a bobbin. More complicated machines called *sergers* use five- or even six-way multithreading in order to simultaneously stitch a seam and wrap thread around it to prevent fraying. Of course, one might argue that because all the threads are just being interleaved by the hardware at an instruction level, this is more akin to “hyperthreading”, but it seems like a debate worth having.

Also, you should definitely cite some of my papers.

¹Just like in GPU computation, come to think of it.



CONFIDENTIAL COMMITTEE MATERIALS

SIGBOVIK 2018 Paper Review

Paper EasierChair: A Survey of Hardware Multithreading

Name Left Blank For Publication

Rating: *6*

Confidence: Unwarranted

Did you know that Jacquard Looms were the first computers? They ran on loops of punch cards, and punch cards and loops are things also associated with computers. Surely any machine that can perform the same sequence of steps over and over – like, say, a player piano or drinking bird – must be Turing complete. Some looms also had counters, and counting is the same as Turing completeness (didn't Alonzo Church show that?).

But let's get back to this great paper. Let me say that I really like it in theory, not that I'd ever do anything with fabric myself. I mean, my inflexible age and gender identity – along with a raft of biases – prevent me from considering that something like this could be enjoyable or useful for people like myself. Let me take a photo for my grandmother/wife/mother/girlfriend/daughter/New Mexico whiptail lizard; she'll love it.

I bought a pair of knit jeans the other day. They were made with an extra-stretchy weave.

How soon will you add this feature to a loom so I can use it to knit a cut-and-sew shirt for my dog? Can you make one from a 3D scan of my body? I just really want to send you a 3D scan of my body. Look, here's a dropbox link written on a napkin.

My t-shirt has a screen print of a houndstooth pattern on it.

Why aren't you doing something that really matters? Y'know, something the economy will reward you for, like extruding useless blobs of plastic, exploiting workers, or selling ephemeral digital tokens?

But seriously, nice work bro, definite publish. See you at my ICO party later?



SIGBOVIK 2018

(Continued) Message from the Organizing Committee

You take your time and carefully follow the floor-2-of-Gates-finding algorithm. As expected of Prestigious Research by Prestigious Researchers, the algorithm works perfectly, and you find yourself on floor 2 of the Gates Hillman Complex. However, SIGBOVIK 2018 does not appear to be located on this floor. After a short search, you find the elevator, by which, remarkably, is a fellow robot! Perhaps they can lead you to SIGBOVIK 2018.

The robot begins to speak: “0100111011—*ahem*, excuse me. Hello, I am CoBot! Can you help me? Could you please press the up elevator button for me?” You oblige. Once in the elevator, CoBot requests, “Can you press the 4 button for me?”

```
switch (choose_dear_reader()) {
case LEND_A_HELPING_PRIMARY_MANIPULATOR:
    “Of course, CoBot! Let’s go to floor 4 together.”
    goto PAGE_124;
case TAKE_THE_SCENIC_ROUTE:
    “How about I press the 3 button, then we take the helix to floor 4?”
    goto PAGE_183;
}
```


Debugging

Amy Likes Spiders

20 COBOLD: Gobblin' up COBOL bugs for fun and profit

squaresLab and Mr. squaresLab SpouseMan

Keywords: software evolution, maintaining software, search-based software engineering, COBOL=\$\$\$\$\$

21 Transactional memory concurrency verification with Landslide

Ben Blum

Keywords: landslide terminal, baggage claim, ground transportation, ticketing

COBOLd: Gobbliⁿ Up COBOL Bugs for Fun and Profit

squaresLab*
Institute for Software Research
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Mr. squaresLab SpouseMan[†]
The Private Sector
Pittsburgh, PA 15213

ABSTRACT

The cost and ubiquity of software bugs has motivated research in automated program repair (APR), a field dedicated to the automatic triage and patching of software defects. APR has typically focused on popular languages such as C and Java, but COBOL developers have suffered six decades of neglect and deserve a repair tool too. We present COBOLd, the first tool for automatically repairing buggy COBOL programs. We demonstrate COBOLd's effectiveness on a COBOL reimplementa-tion of the infamous Zune leap-year bug. We also argue that "we got this" and other researchers should stay away so that we can fix all the COBOL bugs ourselves and thus be filthy, stinking rich.

CCS CONCEPTS

•Software and its engineering →Software evolution; Main-taining software; Search-based software engineering; •COBOL repair →\$;

KEYWORDS

COBOL, SBSE, Software Evolution, Genetic Programming, \$\$\$

ACM Reference format:

squaresLab and Mr. squaresLab SpouseMan. 2018. COBOLd: Gobbliⁿ Up COBOL Bugs for Fun and Profit. In *Proceedings of SIGBOVIK, Pittsburgh, PA USA, March 2018 (SIGBOVIK'18)*, 5 pages. DOI: 10.475/123.4

1 INTRODUCTION

1.1 Automatic Program Repair (APR)

Automatic program repair (APR) is all about fixing bugs automati-cally because software maintenance is expensive. APR techniques take as input (1) a program with a defect and (2) a mechanism to validate correct and incorrect behavior in that program. The val-idation mechanism is typically a test suite: a bug to be repaired corresponds to one or more tests that is failing; one or more pass-ing tests guard against the removal of desirable functionality. The goal is to produce a patch that modifies the original program such

*Chris Timperley, Deby Katz, Zack Coker, Rijnard van Tonder, Mauricio Soto, Afsoon Afzal, Cody Kinneer, Jeremy Lacomis, and Claire Le Goues

[†]Adam Brady

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

SIGBOVIK'18, Pittsburgh, PA USA

© 2018 Copyright held by the owner/author(s). 123-4567-24-567/08/06...\$15.00
DOI: 10.475/123.4

that failing test cases pass without breaking any of the originally passing test cases.

In this work, we automatically fix bugs in COBOL programs using COBOLd, which is just GenProg [10, 11, 17, 18] directly ap-plied to COBOL programs. GenProg uses genetic programming to traverse a space of possible patches to a given buggy program. GenProg uses the test results to define an objective function that informs an evolutionary search [4, 8] over the space of candidate patches. We reuse GenProg because it is well-established and also we know its source code very, very well, having written a large pro-portion of it. It turns out that, given the appropriate configuration options, the code we already have can totally fix bugs in COBOL programs. Who knew?

2 BACKGROUND

Don't know COBOL? Don't worry. This section has all you need.

2.1 COBOL: Is it reasonable?

2.1.1 *Yes.* It is incredibly difficult to create a program that con-tains bugs in COBOL. This is because the design of the language makes it incredibly difficult to write *any* program in COBOL. The lack of advanced features such as *passing parameters to procedures* means that programmers are forced to think very carefully before authoring COBOL programs. Bugs can only exist if programs exist.

2.1.2 *Counterpoint: Absolutely Not.* The first specification of the language (COBOL-60) contained many logical errors and was impossible to interpret unambiguously [2]. This is unsurprising: no academic computer scientists were on the design committee. The committee invented a new metalanguage to define its syntax be-cause no committee members had heard of Backus-Naur form [14].

Dijkstra once declared: "The use of COBOL cripples the mind; its teaching should, therefore, be regarded as a criminal offence." [3]

Also, the "Hello World" program for COBOL is considered non-compliant.¹

2.2 (Inexplicably Recent) Prior Work on COBOL

Unfortunately for Dijkstra, COBOL is still lurking in academic con-ferences and college curriculums (These citations are all real!!! I know. We were also surprised!). Research papers on COBOL in the 21st century cover aspect oriented programming [9, 15] and a (2017!!!) study of code clones [6]. Professors disagreeing with Dijk-stra, or openly admitting to torturing their students, have published

¹The DISPLAY statement (i.e., print) is considered a dangerous pattern according to a COBOL static analyzer: <https://rules.sonarsource.com/cobol/RSPEC-1279>

genetic mutation to generate additional candidates (Line 4) until the oracle condition is satisfied (Line 3). The validation function ORACLE runs the candidate COBOL program on test inputs. If the program produces the expected output, ORACLE returns 1 and we end up with a 100% absolutely correct™ COBOL program. Otherwise, we keep looping until we find a candidate satisfying ORACLE.

3.2 Mutation operators.

COBOLd currently modifies COBOL code by applying one of three different *mutation operators*: *delete*, which deletes a line of code, *add*, which selects a random line of code and inserts a copy of it at a random program point, and *swap*, which exchanges two lines of code. In our exploratory study, these operators were sufficient to generate patches. Future versions of COBOLd will include more COBOL-specific mutation operators such as *delete-a-full-stop*, which randomly deletes a ".", and adaptations, such as only mutating code inside of a procedure.

4 SERIOUSLY: HERE'S AN EXAMPLE

Figure 1 shows a COBOL implementation of the infamous Zune bug [1]. The Microsoft Zune was a popular portable music player that stored the current time as the number of days and seconds since 1 January, 1980. The (buggy) function in Figure 1a was used to compute the current year. An infinite loop occurs when WS-YEAR corresponds to a leap year, and WS-DAYS is less than 366. In this case, the SUBTRACT statement on line 24 is never executed, so the value of WS-DAYS is never changed. COBOLd is able to successfully repair this bug, producing the code shown in Figure 1b. To produce this patch, COBOLd swaps the IF statement on line 23 and the SUBTRACT statement on line 24.

5 WE ARE GOING TO BE VERY, VERY RICH

5.1 No, Really

According to the Cobol Cowboys⁴, COBOL is 65% of active code used today, runs 85% of all business transactions. IBM cites that **200 BILLION** lines of COBOL code is still in use today.⁵ All of Google's code comes in at about 1 percent of this number, at a mere 2 billion lines of code.⁶ This is music to our ears. Rate of adoption for our COBOL repair tool will strictly increase with existing and proliferating COBOL code. Figure 2 conservatively projects our income.

5.2 Treat. Yo. Self

We prepared a shortlist of things to treat ourselves with.

- Clothes
- fragrances
- massages
- mimosas
- fine leather goods
- supercars

⁴<http://cobolcowboys.com/>

⁵Not a joke: <https://www-03.ibm.com/press/us/en/pressrelease/41095.wss>

⁶Not a joke: <https://informationisbeautiful.net/visualizations/million-lines-of-code/>

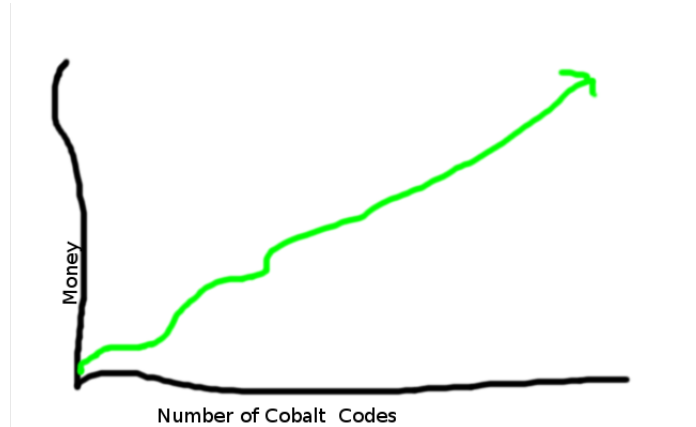


Figure 2: COBOL Repair is going to make us rich.

👁️ 👁️ 👁️ 👁️ 🏆 if i do 5ay so my sel f 🏆 i say so 🏆 thats what im talking about right there right there (chorus: ri ght th ere) mMMMMMM 🏆 👁️ 👁️

It's a modest start, we're new to this. We realize our enormous purchasing power is likely to raise difficult questions. Like, we want private islands too,⁷ but is Australia an island? Is it for sale?

We do plan to use very smol portion of our profits⁸ for the greater good. Our #1 priority is to eradicate video and audio autoplay on all websites. We are willing to pay good money to remove the autoplay attribute from the HTML spec, please get in touch if you know someone. We also plan to resurrect geocities sites. We really miss 88x31 buttons. Click the one below to get the COBOLd demo (or Netscape Now! we're not really sure).



ACKNOWLEDGMENTS

We would like to thank Nalia Soto Gutierrez (Figure 3); Ryan, Kyle (Figure 6), Kevin (Figure 5), and Layla Lacomis (Figure 7); Aries (Figure 12), Cheese (Figure 10), Snickers (Figure 4), and Ampersand LeBrady (Figure 11); Penelope Surden (Figure 8); and Millie Timperley (Figure 9) for all of their love and support, without them this research would not be possible.

We'd also like to more seriously thank Grace Hopper for her work in compilers, for guiding the development of machine-independent programming languages like COBOL, and for being an awesome woman in science. And the ENIAC programmers, who do not get enough recognition.

REFERENCES

- [1] BBC News. 2008. Microsoft Zune affected by 'bug'. In <http://news.bbc.co.uk/2/hi/technology/7806683.stm>.
- [2] Bob Bemer. 1971. A View of the History of COBOL. *Honeywell Computer Journal* (1971).
- [3] Edsger W. Dijkstra. 1975. How do we tell truths that might hurt? (June 1975). published as EWD:EWD498pub.

⁷<https://www.privateislandsonline.com>

⁸which is still going to be like, hundreds of millions

[4] Stephanie Forrest. 1993. Genetic Algorithms: Principles of Natural Selection Applied to Computation. *Science* 261 (Aug. 1993), 872–878.

[5] Mark Harman. 2007. The Current State and Future of Search Based Software Engineering. In *ACM/IEEE International Conference on Software Engineering (ICSE)*. 342–357. <https://doi.org/10.1109/FOSE.2007.29>

[6] Tomomi Hatano and Akihiko Matsuo. 2017. Removing Code Clones from Industrial Systems Using Compiler Directives. In *International Conference on Program Comprehension (ICPC '17)*. 336–345.

[7] Stephen M. Jodis. 1995. Experiences and Successes in Teaching a Language Many CS Students Didn't Want to Learn: COBOL. In *Southeast Regional Conference (ACM-SE 33)*. 273–274.

[8] John R. Koza. 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press.

[9] Ralf Lämmel and Kris De Schutter. 2005. What Does Aspect-oriented Programming Mean to Cobol?. In *International Conference on Aspect-oriented Software Development (AOSD '05)*. 99–110.

[10] Claire Le Goues, Michael Dewey-Vogt, Stephanie Forrest, and Westley Weimer. 2012. A systematic study of automated program repair: Fixing 55 out of 105 bugs for \$8 each. In *AMC/IEEE International Conference on Software Engineering (ICSE)*. Zurich, Switzerland, 3–13.

[11] Claire Le Goues, ThanhVu Nguyen, Stephanie Forrest, and Westley Weimer. 2012. GenProg: A Generic Method for Automatic Software Repair. *IEEE Transactions on Software Engineering (TSE)* 38 (2012), 54–72. <https://doi.org/10.1109/TSE.2011.104>

[12] Ed Lindoo. 2014. Bringing COBOL Back into the College IT Curriculum. *Journal of Computing Sciences in Colleges* 30, 2 (Dec. 2014), 60–66.

[13] Charles R. Litecky and Gordon B. Davis. 1976. A Study of Errors, Error-proneness, and Error Diagnosis in Cobol. *Commun. ACM* 19, 1 (Jan. 1976), 33–38.

[14] Schneiderman. 1985. The Relationship Between COBOL and Computer Science. *Annals of the History of Computing* 7, 4 (1985), 348–352.

[15] Hideaki Shinomi and Yasuhisa Ichimori. 2010. Program Analysis Environment for Writing COBOL Aspects. In *International Conference on Aspect-Oriented Software Development (AOSD '10)*. 222–230.

[16] Shin Hwei Tan and Abhik Roychoudhury. 2015. relifix: Automated Repair of Software Regressions. In *International Conference on Software Engineering (ICSE)*. Florence, Italy.

[17] Westley Weimer, Stephanie Forrest, Claire Le Goues, and ThanhVu Nguyen. 2010. Automatic program repair with evolutionary computation. *Communications of the ACM Research Highlight* 53, 5 (May 2010), 109–116.

[18] Westley Weimer, ThanhVu Nguyen, Claire Le Goues, and Stephanie Forrest. 2009. Automatically finding patches using genetic programming. In *ACM/IEEE International Conference on Software Engineering (ICSE)*. Vancouver, BC, Canada, 364–374. <https://doi.org/10.1109/ICSE.2009.5070536>

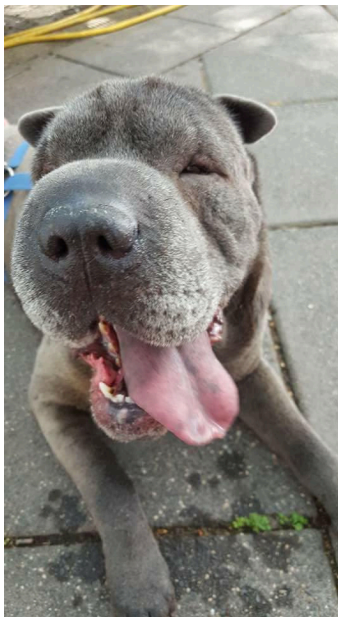


Figure 3: Naila: Was a cat in her previous life. Also, loves pizza more than life itself



Figure 4: Snickers: THE PRETTIEST KITTY



Figure 5: Kevin: Kind of an idiot



Figure 6: Kyle and Ryan: Soft pile



Figure 7: Layla: Actually the prettiest kitty

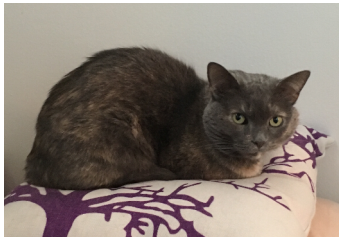


Figure 8: Madam Penelope, Queen of the Apartment

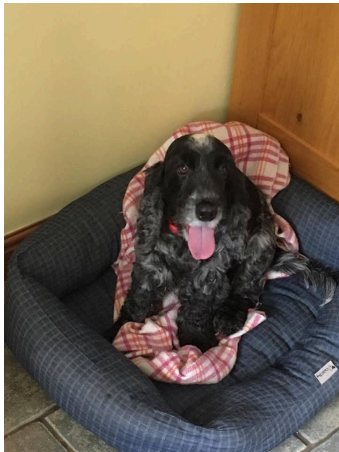


Figure 9: Millie: Stealer of food and hearts

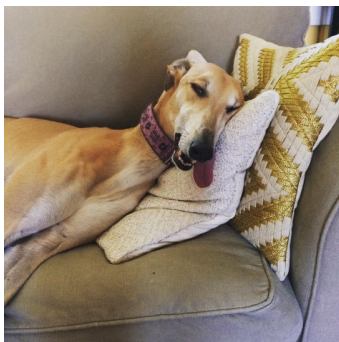


Figure 10: Cheese: SPIRIT OF A CHAMPION



Figure 11: Ampersand: Stone Cold Killer



Figure 12: Aries: Is a bird

Transactional Memory Concurrency Verification with Landslide

Ben Blum

bblum@cs.cmu.edu

Abstract

Hardware transactional memory is a recently-introduced concurrent programming paradigm which allows programmers to elide locks for performance in low-contention workloads. However, it comes at a cost in implementation complexity: fast-path code must be accompanied by backup paths to handle transaction failure. We extend Landslide, a popular stateless model checker, with a concurrency model for transactional memory and evaluate it on several real-world transactional benchmarks and data structure implementations.

Categories and Subject Descriptors D.1.3 [*Programming Techniques*]: Concurrent Programming; D.2.4 [*Software Engineering*]: Software/Program Verification

Keywords landslide terminal, baggage claim, ground transportation, ticketing

1. Introduction

Transactional Synchronization Extensions (TSX) [23] is an instruction set extension for x86 CPUs which adds hardware-based transactional memory. The processor uses its existing cache coherence algorithm to check for memory conflicts with other cores while temporarily staging a sequence of memory accesses. If no other CPU accesses the same memory during the transaction, the access sequence is committed to main memory atomically (with respect to visibility by other CPUs). Otherwise, the accesses are discarded, the CPU's local state is reverted, and the transaction returns a failure code.

This feature can be used to replace conventional locking in performance-critical concurrent programs. When a concurrent workload accesses largely thread-local data, or disjoint sections of a shared data structure, the contention rate between threads is low, and transactions will often succeed. Compared to programs which use conven-

tional locks, which use bus-locking atomic accesses even in the fastest code path, TSX provides substantial performance improvements in such programs [10, 12, 44]. However, the possibility of transaction failure introduces additional implementation complexity: programmers must also provide a backup plan to safely resolve contention between threads, usually involving conventional synchronization. These backup paths must coordinate not only with other backup paths but also with other fast paths which another thread may begin after the original transaction failed, which even in the simplest transactions requires complex synchronization sequences [10]. This introduces an additional dimension of nondeterminism into an already concurrent program, and moreover, because transaction failure is expected to be rare, obscure interleavings between failure paths are difficult to expose during stress testing.

This motivates the use of stateless model checking (MC) [19] to comprehensively verify these transactional programs, fast paths failure paths and all. MC aims to force the system to execute all possible thread interleavings under a given test case, exhaustively checking for bugs or verifying their absence in the corresponding state space. Many such model checkers exist, varying in interleaving granularity, memory analysis, types of programs checked, and search ordering strategy [9, 24, 25, 27, 28, 30, 32, 38, 43]. This work builds upon Landslide [4], a simulator-based tester which checks both user- and kernel-level programs and incorporates data-race analysis [16, 36] to find new preemption points at memory access granularity. Our contributions are as follows:

1. We extend Landslide's concurrency model to include transaction failure as an additional source of nondeterminism;
2. We provide a proof sketch that our implementation matches TSX's execution semantics,
3. We evaluate the extended Landslide on several transactional programs, analyzing both its bug-finding and verification performance.

The paper is organized as follows. Section 1 introduces the problem domain and motivates our research. The other sections state the rest of the paper.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee, provided... honestly, provided nothing. The ACH is already flattered enough that you're even reading this notice. Copyrights for components of this work owned by others than ACH must be laughed at, then ignored. Abstracting with credit is permitted, but abstracting with cash is preferred. And please tip us, for the love of Turing.

SIGBOVIK '18 Pittsburgh, PA, USA
Copyright © 2018 held by owner/author(s). Publication rights licensed to ACH.
ACH...\$15.00

2. Background

This section introduces the fundamental concepts and prior work in both hardware transactional memory and stateless model checking, which we propose to combine.

Hardware transactional memory. TSX was implemented on consumer hardware for the first time by Intel’s Haswell architecture [22], which extends the x86 instruction set to provide `xbegin`, `xend`, and `xabort` for beginning, committing, and aborting transactions, respectively. Higher-level programming languages or compilers may offer libraries or intrinsics to access these instructions; for C and C++ GCC provides intrinsics named `_xbegin()` and so on [18]. Figure 1 shows an example program using TSX to synchronize access to a shared counter, including a failure path which defaults to a conventional lock. This example actually has a bug, which we will discuss in the next section; the reader is encouraged to try to spot it before then. Several related works formally prove the correctness of transactional memory *implementations* [13, 20, 21, 34], but verifying the client programs written to use transactions remains an open problem.

Under software transactional memory (STM) [2], memory conflicts with other threads are the only reason for transaction failure (apart from programmer-supplied explicit aborts); hence, depending on program semantics, some transactions may be guaranteed to succeed. However, hardware transactions (HTM) may also fail for several other reasons such as random system interrupts or exhausting the CPU’s cache capacity. Because timer interrupts can in principle occur at any moment, and with arbitrary frequency (observable by the program, perhaps as a result of a heavily-loaded system), in this paper we will simplify the failure model by saying that HTM transactions can fail for any reason. We defer discussion of programs which distinguish the reason for aborts through the failure code to Section 5.

Stateless model checking. Model checking (MC) [19] is a testing technique for systematically executing and verifying the possible thread interleavings of a concurrent program. The main research challenge is to cope with exponential explosion of the state space, which is sized $O(n^k)$ for a program with n operations and k threads. Some *stateful* MCs explicitly store and compare all visited states of the program being tested [24], which both keeps track of test coverage and allows identifying identical states to avoid testing redundant interleavings. By contrast, *stateless* MC (henceforth abbreviated simply as MC) stores only the current sequence of execution events to avoid a prohibitive memory footprint. Reduction algorithms [1, 11, 17, 25, 25, 45] can then analyze the memory accesses in that sequence to identify interleavings observationally-equivalent under Mazurkiewicz trace theory [31] and hence safe to skip. The resulting state spaces are still exponentially-sized, but only in the number of

```
1  if ((status = _xbegin()) == SUCCESS) {
2      x++;
3      _xend();
4  } else {
5      mutex_lock(&m);
6      x++;
7      mutex_unlock(&m);
8  }
```

Figure 1. Example transactional program. If the top branch aborts, execution will revert to the return of `_xbegin()` and control will drop into the `else` branch. The programmer can then use explicit synchronization to resolve the conflict.

conflicting operations rather than all operations. Of these, Landslide uses Dynamic Partial Order Reduction (DPOR) [17] to prune its state spaces.

MCs may instrument programs to introduce thread switches at varying granularity, which affects the number of operations n . Some target distributed systems, instrumenting only message-passing events [42]; some run multithreaded programs natively, instrumenting only the pthread API for performance [38]; and some insert compiler instrumentation on statically-identified memory accesses [32, 43]. Landslide traces every memory access through the use of a simulated environment [29], which is important for identifying data races to use as new pre-emption points [8], as well as for identifying when a memory conflict may cause transaction aborts. With regard to checking for bugs, the “model” the name refers to being checked may be an external formal specification, the program’s own internal consistency checks, or a set of expected properties encoded in the tool itself. Landslide uses the latter two cases, checking for assertion failures as well as deadlocks, use-after-frees, and segfaults. For this work we also detect use of `xend` outside of a transaction or `xbegin` within one as a bug.

3. Design

This section presents our formalization of transactional memory in Landslide’s framework of thread concurrency. We make two major simplifications: simulating transaction aborts as immediate failure injections, and treating transaction atomicity as a global mutex during data-race analysis; and provide corresponding equivalence proofs.

Notation. Let $I = TN_1@L_1, TN_2@L_2, \dots, TN_n@L_n$, with N_i a thread ID and L_i a code line number, denote the execution sequence of a program as it runs according to the specified thread interleaving.¹

¹This serialization of concurrent execution is told from the perspective of all CPUs at once and hence assumes sequential consistency. For discussion of relaxed memory models refer to Section 5.

3.1 Example

Consider again the program in Figure 1. Note that the C-style `x++` operations, when compiled into assembly [41], results in multiple memory accesses which can be interleaved with other threads.

```
2a temp <- x;
2b temp <- temp + 1;
2c x <- temp;
```

If these instructions from the `x++` in the transaction are preempted, with another thread's access to `x` interleaved in between, the transaction will abort. So, the interleaving

$T1@1, T1@2a, T1@2b, T2@1, T2@2, T2@3, T1@2c, T1@3$

or, henceforth abbreviated for clarity:

$T1@1-2b, T2@1-3, T1@2c-3$

is not possible; rather, $T1$ will fall into the backup path:

$T1@1-2b, T2@1-3, T1@4-7$

However, the `x++` operation from the failure path (correspondingly 6a, 6b, 6c) can be thusly separated with conflicting accesses interleaved in between, since the mutex only protects the failure path against other failure paths, but not against the transaction itself. So (assuming `x` is intended to be a precise counter rather than a sloppy one), we observe a bug in the following interleaving.²

$T1@1-2b, T2@1-3, T1@4-6b, T3@1-3, T1@6c-7$

Prior work [10] proposed the idiom shown in Figure 2 to exclude this family of interleavings, which shows that correctly synchronizing even the simplest transactions may be surprisingly difficult or complex.

3.2 Modeling Transaction Failure

Left unstated in interleavings such as $T1@1-2c, T2@1-3, T1@4-7$ ³ are HTM's execution semantics, namely:

1. any modifications to shared state (such as 2c) by $T1$ are not visible to $T2$ during its execution, despite $T2$ being executed afterwards, and
2. all local and global state changes by $T1$ between lines 1 and 2c are discarded when jumping to line 4.

While use of TSX in production requires the performance advantage of temporarily staging such accesses in local CPU cache, model checking such programs need be concerned only with the program's *observable* behaviours. We

²Note also that this bug requires either at least 3 threads or at least 2 iterations between 2 threads to expose; this highlights MC's dependence on its test cases to produce meaningful state spaces in the first place.

³For a clearer example we reorder $T1$'s write to `x` before $T2$'s part here.

```
prevent_transactions = false;
0 while (prevent_transactions) continue;
1 if ((status = _xbegin()) == SUCCESS) {
2     if (prevent_transactions)
3         _xabort();
4     x++;
5     _xend();
6 } else {
7     mutex_lock(&m);
8     prevent_transactions = true;
9     x++;
A    prevent_transactions = false;
B    mutex_unlock(&m);
C }
```

Figure 2. Variant of the program in Figure 1, with additional synchronization to protect the failure path from the transactional path. The optional line 0 serves to prevent a cascade of failure paths for the sake of performance by allowing threads to wait until transacting is safe again.

claim that MCing the simpler interleaving $T1@1, T2@1-3, T1@4-7$ is an equivalent verification as MCing the one above; in fact, this interleaving suffices to check all observable behaviours of all interleavings of all subsets of $T2@1-3$ with all subsets of $T1@2a-2c$, whether they share a memory conflict or not. Stated formally:

Lemma 1 (Equivalence of Aborts). *Let:*

- $Ti@α$ be an HTM begin operation,
- $Ti@β_1 \dots Ti@β_n$ be the transaction body (with $β_n$ the HTM end call),
- $Ti@φ_1 \dots Ti@φ_m$ be the failure path, and
- $Ti@ω_1 \dots Ti@ω_l$ be the subsequent code executed unconditionally.⁴

Then, for any interleaving prefix⁵

$Ti@α, Ti@β_1 \dots Ti@β_b,$
 $Tj@γ_1 \dots Tj@γ_j,$
 $Tk@κ_1 \dots Tk@κ_k,$
 $Ti@β_{b+1}$

with $b < n, j \neq i, k \neq i$, etc., either:

1. $Ti@α, Tj@γ_1 \dots Tj@γ_j, Tk@κ_1 \dots Tk@κ_k, Ti@φ_1 \dots$ (conflicting case), or
2. $Ti@α, Ti@β_1 \dots Ti@β_b \dots Ti@β_n, Tj@γ_1 \dots Tj@γ_j, Tk@κ_1 \dots Tk@κ_k$ (independent case)

exists and is observationally equivalent.

⁴Arbitrary code may not be structured to distinguish these as nicely as in our examples; e.g., more code may exist in the success branch after `_xend()`; such would be considered part of $ω$ here.

⁵Without loss of generality: for any number of other threads Tj/Tk , and for any number of thread switches away from Ti during the transaction.

Proof Sketch. We case on whether the operations by \mathbf{Tj} and/or \mathbf{Tk} have any memory conflicts (read/write or write/write) with $\mathbf{Ti@}\beta_1 \dots \mathbf{Ti@}\beta_n$. If so, then the hardware will abort \mathbf{Ti} 's transaction, discarding the effects of $\mathbf{Ti@}\beta_1 \dots \mathbf{Ti@}\beta_n$ and jumping to $\mathbf{Ti@}\phi_1$, satisfying case 1. Otherwise, by DPOR's definition of transition dependence [17], $\mathbf{Ti@}\beta_{b+1} \dots \mathbf{Ti@}\beta_n$ is independent with the transitions of \mathbf{Tj} and \mathbf{Tk} , may be successfully executed until transaction commit, and reordering them produces an equivalent interleaving, satisfying case 2. \square

The second part of our claim follows naturally.

Theorem 1 (Atomicity of Transactions). *For any state space S of a transactionally-concurrent program, an equivalent state space exists in which all transactions are either executed atomically or aborted immediately.*

Proof Sketch. For every $I \in S$ with $\mathbf{Ti@}\alpha, \mathbf{Ti@}\beta_1 \dots \mathbf{Ti@}\beta_b, \mathbf{Tj@}\dots, \mathbf{Tk@}\dots, \mathbf{Ti@}\beta_{b+1} \in I$, apply Lemma 1 to obtain an equivalent interleaving I' satisfying the theorem condition. The resulting S' can then be MCed without ever simulating HTM rollbacks. \square

3.3 Memory Access Analysis

Next, we address the memory accesses within transactions with regard to data-race analysis. From Theorem 1 we have that the body of all transactions may be executed atomically within the MC environment. While they may interleave between other non-transactional sequences, no other operations (whether transactional or not) will interrupt them. We claim this level of atomicity is equivalent to that provided by a global lock, and hence abstracting it as such in Landslide's data-race analysis is sound.

Let $\mathbf{Ti@}\mu, \mathbf{Tj@}\nu$ be a pair of memory accesses to the same address, at least one a write, in some transactional execution I normalized under Lemma 1. Then let $\text{lockify}_m(\mathbf{Tk@}L)$ denote a function over instructions in I , which replaces $\mathbf{Tk@}L$ with $\mathbf{Tk@lock}(m)$ if L is a successful HTM begin, with a no-op if L is a transaction abort, or with $\mathbf{Tk@unlock}(m)$ if L is an HTM end, or no replacement otherwise. Finally, let $I' = \exists m. \text{lockify}_m(I)$, the execution with the boundaries of all successful transactions replaced by an abstract global lock. Lemma 1 guarantees mutual exclusion of m .

Theorem 2 (Transactions are a Global Lock). $\mathbf{Ti@}\mu, \mathbf{Tj@}\nu$ is a data race in I iff it is a data race in I' .

Proof Sketch. We prove one case for each variant definition for data races supported in Landslide [8]. For each, we semiformally state what it means to race in an execution with synchronizing HTM instructions.

- **Limited Happens-Before.** To race in I they must be reorderable at instruction granularity, at least one with a thread switch immediately before or after. [33, 36].

- $I \Rightarrow I'$: If $\mathbf{Ti@}\mu, \mathbf{Tj@}\nu$ race in I , then they cannot both be in successful transactions, or else placing $\mathbf{Ti@}\mu$ within the boundaries of $\mathbf{Tj@}\nu$'s transaction would cause the latter to abort, invalidating $\mathbf{Tj@}\nu$, or vice versa. Hence they will not both hold m in I' . Otherwise their lock-sets and DPOR dependence relation remain unchanged.

- $I' \Rightarrow I$: If $\mathbf{Ti@}\mu, \mathbf{Tj@}\nu$ race in I' , both corresponding threads cannot hold m ; WLOG let \mathbf{Ti} not hold m during $\mathbf{Ti@}\mu$. Then in I , $\mathbf{Ti@}\mu$ is not in a transaction. With the remainder of their lock-sets still disjoint, and still not DPOR-dependent, $\mathbf{Tj@}\nu$ (or its containing transaction) can then be reordered directly before or after $\mathbf{Ti@}\mu$.

- **Pure Happens-Before.** WLOG fix $\mathbf{Ti@}\mu < \mathbf{Tj@}\nu \in I$. Then to race in I there must be no pair of synchronizing instructions $\mathbf{Ti@}\epsilon$ (a release edge) and $\mathbf{Tj@}\chi$ (an acquire edge) such that

$$\mathbf{Ti@}\mu < \mathbf{Ti@}\epsilon < \mathbf{Tj@}\chi < \mathbf{Tj@}\nu \in I$$

to update the vector clock epoch between $\mathbf{Ti@}\mu$ and $\mathbf{Tj@}\nu$ [16, 35].

- $I \Rightarrow I'$: If $\mathbf{Ti@}\mu, \mathbf{Tj@}\nu$ race in I , then they cannot both be in successful transactions, or else Lemma 1 normalization would provide the corresponding HTM end and begin for $\mathbf{Ti@}\epsilon$ and $\mathbf{Tj@}\chi$ respectively. Hence there will be no unlock/lock pair on m in I' to satisfy the above sequence.

- $I' \Rightarrow I$: If $\mathbf{Ti@}\mu, \mathbf{Tj@}\nu$ race in I' , then they cannot both hold m , or else lockify_m would provide the corresponding unlock and lock for $\mathbf{Ti@}\epsilon$ and $\mathbf{Tj@}\chi$ respectively. Hence there will be no HTM end/begin pair in I to satisfy the above sequence.

Hence, data-race analysis is sound when transaction boundaries are replaced by an abstract global lock. \square

3.4 Implementation

Our implementation of HTM-equivalent semantics has been incorporated by the Landslide maintainers upstream. It is available open-source at <https://github.com/bblum/landslide>. Programs should be ported to the Pebbles userland [14, 15], their use of compiler HTM intrinsics should be replaced with the Landslide stubs provided in `410user/inc/htm.h`, and HTM nondeterminism can then be enabled with the `-X` command-line flag. We have also extended its Iterative Deepening implementation [8] with a new option (`-M` flag) to optimize for completion time by prioritizing the maximal state space job and cancelling all others (which maintains its soundness guarantee) which we will use in our evaluation. All test cases therein are also available in the repository linked above.

buggy test	params	Quicksand mode			Maximal state space mode (-M)			
		cpu (s)	wall (s)	int's	cpu (s)	wall (s)	int's	SS size (est.)
htm1 (assertion)	2,1	45.78	9.70	21	*9.47	*6.40	21	213
	2,2	84.14	13.59	*33	*10.39	*7.70	49	1536
	2,3	131.91	20.44	*73	*12.83	*9.67	113	10752
	2,4	255.75	37.56	257	*18.63	*15.86	257	73728
	3,1	114.06	17.45	*15	*9.50	*6.79	21	13653
	3,2	109.60	26.16	49	*10.72	*7.97	49	393216
	3,3	124.80	20.40	*73	*13.84	*11.01	113	11010048
	3,4	227.49	35.15	*161	*31.37	*28.53	257	301989888
	4,1	53.08	9.79	*15	*9.82	*7.00	21	873813
	4,2	117.07	19.09	*33	*11.54	*8.55	49	100663296
swapbug (deadlock)	2,1	70.95	13.45	*16	*38.96	*13.15	109	194
	2,2	107.28	*17.45	*146	*44.73	19.47	281	1620
	2,3	280.05	38.70	*352	*60.30	*35.55	718	12748
	2,4	617.94	*81.50	*834	*108.58	82.60	1820	97823
	3,1	*1275.04	*163.42	*771	-	>30m	-	184984
	3,2	-	>30m	-	-	>30m	-	3099225
avl_insert (segfault+)	2,2	488.07	64.77	*83	*81.00	*40.30	336	379982
	2,3	2670.87	*330.45	*3066	*1331.79	1274.36	13926	96248131
	2,4	*3259.37	*436.50	*1639	-	>30m	-	36019973
	3,1	222.02	40.04	*28	*69.99	*24.25	78	1572107
	3,2	*1569.09	*216.85	*209	-	>30m	-	1402363529

Table 1. Landslide’s bug-finding performance on various test configurations. Quicksand’s workqueue approach optimized for fast bug-finding is compared against our maximum-state-space-prioritizing approach for fast verification. For each, we list the CPU-time and wall-clock time elapsed, plus the number of interleavings of the ultimately buggy state space tested, before the bug was found. * marks the winning measurements between each series. Lastly, Landslide’s state space size estimation [39], though approximate at best, confers a sense of the exponential explosion.

4. Evaluation

To the best of our knowledge, this is the first work to test transactional programs in a model-checking environment, so no other MC State of the Art⁶ exists to compare to in controlled experiments. Nevertheless, we pose the following evaluation questions.

1. How quickly does Landslide find bugs in incorrect transactional programs of varying sizes?
2. How quickly does Landslide verify correct transactional programs of varying sizes?
3. By the way, should MC research papers quantify variance in their CPU-time performance experiments?

Our evaluation suite comprises several hand-written unit tests and [10]’s microbenchmarks and transactional AVL tree and separate-chaining hashmap, as follows.

- **htm1**: The bug from Figure 1.
- **htm2**: The fixed version as in Figure 2.
- **counter**: Microbenchmark version of **htm2** which replaces the complex failure path with a simple `xadd`.

- **swap**: Microbenchmark that swaps values in an array.
- **swapbug**: **swap** modified to introduce circular locking in the failure path.
- **avl_insert**: AVL tree concurrent insertion test.
- **avl_fixed**: **avl_insert** with the AVL bug fixed (spoilrs!!).
- **map_basic**: Separate-chaining hashmap concurrent insertion test.
- **map_basicer**: **map_basic** modified with a larger initial size to skip the resizing step.

The notation `testname(K,N)` will denote a test configuration of K threads, each running N iterations of the test logic. All tests were run on an 8-core 2.7GHz Core i7 with 32 GB RAM.

4.1 Bugs

Table 1 presents our bug-finding results. We configured Landslide to run the Quicksand algorithm [8] shown left, as well as to prioritize the maximal state space as discussed above, shown right, each with a time limit of 30 minutes. We draw three main conclusions from this data.

⁶The author’s DJ name.

```

while (_retry);
if (_xbegin() == SUCCESS) {
    tie(_root,inserted) = _insert(_root,n);
    _xend();
} else {
    pthread_mutex_lock(&_tree_lock);
    _retry = true;
    tie(_root,inserted) = _insert(_root,n);
    _retry = false;
    pthread_mutex_unlock(&_tree_lock);
}

```

Figure 3. Unmodified code from `htmaavl.cpp` showing the previously-unknown bug found by Landslide. The transaction path fails to check `_retry`, leading to data races and corruption just as in `htm1`.

Finding bugs quickly. As the test parameters increase, the multiplicative factor in bug-finding speed (2-4x, eyeballing) is generally smaller than that of the total number of interleavings (10-100x). In other words, should they exist, Landslide find bugs reasonably quickly in these transactional programs despite prohibitive exponential explosion in total state space size. This corroborates the prior work [8], extending its good news to the world of HTM.

New bugs. In addition to the bugs we intentionally wrote in `htm1` and `swapbug`, to our pleasant surprise Landslide also found a previously-unknown bug in the transactional AVL tree, exposed by `avl_insert` with any parameters higher than (2,1). Figure 3 shows the root cause, essentially the `htm1` bug in disguise. This manifested alternately as a segfault and as a consistency-check assertion failure. The presence of `while (_retry);` makes the necessary preemption window extremely small in practice (between it and `_xbegin()`), whereas MC is blind to such matters of chance.⁷ Moreover, we conclude that even Figure 2’s protocol’s very proposer getting it wrong motivates the need for MC on such programs, and suggests TSX primitives should be encapsulated behind higher-level abstractions such as lock elision [26], which can be verified in isolation with smaller state spaces than trusted in turn when checking their client programs [37].

Performance comparison. Quicksand’s ability to find bugs in fewer distinct interleavings does not necessarily correlate with better performance. Most of our tests are too small for its approach to pay off, with `swapbug(3,1)` and `avl_insert` as its notable wins. While the prior work showed plenty more wins [8], future MCs could prioritize state spaces using not just size estimation but state space maximality as well to soften the trade-off both for smaller tests and for verification. Speaking of which...

⁷ Considering the loop does not affect the test’s possible behaviours, only its *likely* ones, we removed it in Landslide’s version of the test to keep the state space size manageable.

4.2 Verification

Landslide proved the following tests correct. With no bugs to find, comparing the different testing modes against each other is meaningless; we simply present their state space sizes and runtime (using `-M`) in Table 2.

test	params	cpu (s) (or †ETA)	SS size (or †est.)
htm2	2,1	18.57	294
	2,2	133.78	4902
	2,3	1986.98	79017
	3,1	11672.15	467730
	3,2	†10d 14h	†13763999
counter	2,1	5.57	30
	2,2	15.53	384
	2,3	155.00	5280
	2,4	2211.10	75264
	3,1	57.90	1960
3,2	10028.93	329888	
swap	2,1	32.98	193
	2,2	3652.91	101150
	3,1	†8d 12h	†411312
avl_insert	2,1	1083.35	40062
avl_fixed	2,1	1079.03	45078
	2,2	†2762y	†8714863
	3,1	†12d 2h	†11498545
map_basic	2,1	†10d 17h	†16388977
map_basicer	2,1	877.44	28635
	2,2	†2d 7h	†5925634
	3,1	†468d 13h	†35893653

Table 2. Transactional tests verified (or not) by Landslide.

Several larger tests we were unable to complete before the submission deadline are also shown. These are indicated with † and their listed ETAs and state space sizes represent Landslide’s estimate after a timeout of 1 hour (hopefully enough for data-race PPs to saturate and estimates to stabilize, although note the two estimate types use different algorithms so may disagree significantly). Though these tests proved beyond our reach, in contrast with the instant gratification expected of bug-finding, the verification guarantee may in some cases be worth the estimated time requirement for widely-used industrial implementations. Future work may also expand our coverage upon this frontier [7].

4.3 Variance

Because many of the verification tests are long-running, and we are writing this too close to the submission deadline (who doesn’t), we regret being unable to present every performance measurement above as an average of multiple samples with error bars [40]. Nevertheless, we make some effort to address variance.

We noticed significant slowdowns when varying several aspects of our experimental environment. For example, multiple Landslides running at once slow each other down, likely arising from kernel resource contention as Landslide uses `fork()` to save simulation state. Table 3 shows the impact of running a single Landslide instance with `-M` on `counter(2,2)` with various programs also running, despite never saturating our test system’s 8 CPUs.

Table 4 shows the impact of Landslide needing to automatically annotate `counter(2,2)` being run for the first time, rather than reusing existing instrumentation, as well as the variance of Quicksand mode. Because Quicksand repeats more work across jobs, rather than comparing it to a baseline, we show in the right column how many total interleavings each approach executed. (The maximal state space alone comprises 384 in all cases.) The quicksand variance was surprisingly bimodal, with 6 samples in a distribution of 106.14 ± 1.22 and 4 in 142.11 ± 1.67 , suggesting two distinct scheduling patterns for its workqueue threads. Future work should figure out why.

We conclude that MC performance evaluations must address experimental environment variables to ensure consistent performance between runs. So doing, multiple samples and error bars are then necessary only when using nondeterministic search ordering strategies such as Iterative Deepening. Otherwise, considering the low variance shown in Table 3, they need be shown only for a token small test to provide reader some assurance of similar consistency in the larger tests (especially if the time tradeoff to measure them would sacrifice testing larger state spaces to begin with).

For all tests outside of Table 3, we fixed our environment by measuring all performance numbers with Firefox and Chrome as the only other significant machine load. We believe the exponential differences among completion times justifies the absence of error bars, which one would expect to show 2% variance if extrapolating from these results. The numbers of interleavings in each state space are, of course, deterministic and do not vary across runs.

5. Limitations

This section should be mandatory in all systems papers.

Transaction failure codes. When a transaction fails, `_xbegin()` returns a failure code denoting the reason, or combination thereof, therefor [18]. If a program then cases on that failure code to select between different backup paths, model checking it by simply injecting a single type of failure may be unsound. For example, a program executing a transaction which is guaranteed to never conflict with any other threads, and hence never abort without `_XABORT_RETRY`, could legally `assert(false)`; in its failure path, while our approach in Section 3 would erroneously trigger that assertion and report a bug. Likewise, the transactional data structures

load	cpu (s)	vs self avg	vs baseline
none	14.95 ± 0.17	0.99-1.02x	(baseline)
vid-L	15.13 ± 0.10	0.99-1.01x	1.01x
ff/c	15.55 ± 0.16	0.99-1.02x	1.04x
sm5	16.63 ± 0.07	0.99-1.01x	1.11x
ff/c+vid-S	17.09 ± 0.34	0.98-1.05x	1.14x
ls	19.11 ± 0.32	0.97-1.03x	1.28x
ff/c+ls	19.66 ± 0.50	0.96-1.02x	1.31x

Table 3. Performance variance on `counter(2,2)` with other programs running on the test machine. `vid-L` is full-screen video (played locally with `mpPlayer`), `ff/c` is Firefox and Chrome (idle, ≤ 20 tabs), `sm5` is StepMania 5.1 (during gameplay [5]), `vid-S` is full-screen video (streamed via Crunchyroll), `ls` is a 2nd instance of Landslide. Average of 10 samples, $\pm N$ is 1 stdev.

Landslide mode	cpu (s)	total int’s
verif (-M)	14.95 ± 00.17	403
reinstrument	24.72 ± 00.10	403
quicksand (no -M)	120.53 ± 18.62	2128

Table 4. Performance variance on `counter(2,2)` in various modes. Average of 10 samples, $\pm N$ is 1 stdev.

from [10] abstract away any spurious `_XABORT_RETRY` aborts behind a retry loop; a MC unwise to that idiom would call it an infinite loop bug.

Our HTM implementation includes an experimental feature to track the set of abort codes possible for each transaction. `_XABORT_RETRY` is always enabled; then, it harnesses DPOR’s existing memory analysis to identify when `_XABORT_CONFLICT` is possible, and instruments `_xabort()` calls to record any user-supplied codes for `_XABORT_EXPLICIT`. This comes at a cost of even more state space explosion, increasing the exponent at each `_xbegin()` preemption point by (usually) 1 plus however many distinct `_xabort()` codes the program uses.

Many such branches may be equivalent; for example, explicit and conflict aborts need not be tested separately in transactions whose failure paths do not distinguish the cause, and the perennial `_XABORT_RETRY` abort can be skipped entirely if the client abstracts it away behind a retry loop. In fact, applying both reductions simultaneously amounts to the STM concurrency model [2]⁸ and may reduce the state space even smaller than the original. Testing them by hand⁹ reduced `map_basicer(2,1)`’s 28635 interleavings to 11577, and produced the same 384-interleaving state space on `counter(2,2)` (in which DPOR triggers conflict aborts on every transaction any-

⁸ Proof left to future work. We’ve proved enough here already.

⁹ Using visual inspection of the test to trust the reductions’ soundness.

way). Future work could use static or dynamic flow analysis to identify such reduction opportunities automatically; for now, this feature is disabled by default but accessible via the `-A` command-line option (in addition to `-X`).

Relaxed memory orderings. Section 3’s formalization of thread interleavings does not account for read/write reorderings possible on relaxed consistency architectures [3]. In fact, even after [10]’s proposed fix, our running example program is still incorrect on Total Store Order (TSO) architectures such as x86. Despite stores being totally-ordered, x86 may still reorder stores after subsequent loads. Accordingly, an execution of `8, 9a, 9b, 9c` in Figure 2 may be locally visible to another thread as `9a, 8, 9b, 9c`, and hence an apparent interleaving of

`T1@1, T2@1–5, T1@7, T1@9a, T3@1–5, T1@8, T1@9b–B`

is possible (reordered accesses underlined for emphasis). An `mfence` barrier is needed between lines 8 and 9 to solve this problem on TSO [6]. On Partial Store Order (PSO) architectures, even more barriers may be necessary.

Because Landslide’s concurrency model includes only instruction-level thread nondeterminism, not per-CPU memory buffer reorderings, our current HTM implementation cannot find this bug. In fact, it erroneously verifies the corresponding test `htm2(3,1)` in 3 CPU-hours, with 467730 distinct interleavings in total, none of which include the above-listed sequence. Recent work extended DPOR to support TSO and PSO memory nondeterminism [45]; if incorporated into Landslide, we could find or verify the absence of such bugs. Visual inspection of [10]’s HTM data structures found no barriers used in this implementation pattern; we would urge any reader interested in using those to add them in by hand first.

6. Conclusion

Stateless model checking research is a perpetual existence of staring up the sheer cliff face that is the exponential curve. As new concurrency paradigms emerge, we make our living by adapting our reduction algorithms and search strategies to them to climb ever higher on that curve, eking out a few more loop iterations or slightly higher thread counts in our verification guarantees. Whether that is beautiful in its imperfection or cause for despair is merely a matter of perspective. Also, the author hopes their committee won’t mind the publication venue when they cite this in their thesis.

Acknowledgments

We thank Mario Dehesa-Azuara for generously providing the HTM data structures used in our evaluation, the anonymous SIGBOVIK program committee, and anyone who actually read this unusual paper all the way through. This work was supported in part by the U.S. Army Research Office under grant number W911NF0910273.

References

- [1] P. Abdulla, S. Aronis, B. Jonsson, and K. Sagonas. Optimal dynamic partial order reduction. In *Principles of Programming Languages*, POPL ’14. ACM, 2014.
- [2] A.-R. Adl-Tabatabai, B. T. Lewis, V. Menon, B. R. Murphy, B. Saha, and T. Shpeisman. Compiler and runtime support for efficient software transactional memory. In *Programming Language Design and Implementation*, PLDI ’06. ACM, 2006.
- [3] S. V. Adve and K. Gharachorloo. Shared memory consistency models: A tutorial. *Computer*, 29(12), Dec. 1996.
- [4] B. Blum. Landslide: Systematic dynamic race detection in kernel space. Master’s thesis, Carnegie Mellon University, May 2012.
- [5] B. Blum. A boring follow-up paper to “Which ITG stepcharts are turniest?” titled, “Which ITG stepcharts are crossoveriest and/or footswitchiest?”. In *Conference in Celebration of Harry Q. Bovik’s 26th Birthday*, SIGBOVIK ’17. ACH, 2017.
- [6] B. Blum. Demonstration of the need for a barrier in TSX failure paths. <https://gist.github.com/bblum/85f64858a35a74641be228f191144911>, 2018.
- [7] B. Blum. *Practical Concurrency Testing or, How I Learned to Stop Worrying and Love the Exponential Explosion*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 2018.
- [8] B. Blum and G. Gibson. Stateless model checking with data-race preemption points. In *Object-Oriented Programming, Systems, Languages, and Applications*, OOPSLA 2016. ACM, 2016.
- [9] S. Burckhardt, P. Kothari, M. Musuvathi, and S. Nagarakatte. A randomized scheduler with probabilistic guarantees of finding bugs. In *Architectural Support for Programming Languages and Operating Systems*, ASPLOS XV. ACM, 2010.
- [10] M. Dehesa-Azuara and N. Stanley. Hardware transactional memory with Intel’s TSX. <http://www.contrib.andrew.cmu.edu/~mdehesaa/>, 2016.
- [11] B. Demsky and P. Lam. SATCheck: SAT-directed stateless model checking for SC and TSO. In *Object-Oriented Programming, Systems, Languages, and Applications*, OOPSLA 2015. ACM, 2015.
- [12] D. Dice, Y. Lev, M. Moir, and D. Nussbaum. Early experience with a commercial hardware transactional memory implementation. In *Architectural Support for Programming Languages and Operating Systems*, ASPLOS XIV. ACM, 2009.
- [13] S. Doherty, L. Groves, V. Luchangco, and M. Moir. Towards formally specifying and verifying transactional memory. *Electron. Notes Theor. Comput. Sci.*, 259, Dec. 2009.
- [14] D. Eckhardt. Pebbles kernel specification. <http://www.cs.cmu.edu/~410-s18/p2/kspec.pdf>, 2018.
- [15] D. Eckhardt. Project 2: User level thread library. http://www.cs.cmu.edu/~410-s18/p2/thr_lib.pdf, 2018.
- [16] C. Flanagan and S. N. Freund. FastTrack: Efficient and precise dynamic race detection. In *Programming Language Design and Implementation*, PLDI ’09. ACM, 2009.

- [17] C. Flanagan and P. Godefroid. Dynamic partial-order reduction for model checking software. In *Principles of Programming Languages*, POPL '05. ACM, 2005.
- [18] GNU Foundation. X86 transaction memory intrinsics. <https://gcc.gnu.org/onlinedocs/gcc-4.8.2/gcc/X86-transactional-memory-intrinsics.html>, 2016.
- [19] P. Godefroid. VeriSoft: A tool for the automatic analysis of concurrent reactive software. In *Computer Aided Verification*, CAV '97. Springer-Verlag, 1997.
- [20] R. Guerraoui, T. A. Henzinger, and V. Singh. Completeness and nondeterminism in model checking transactional memories. In *Concurrency Theory*, CONCUR '08. Springer-Verlag, 2008.
- [21] R. Guerraoui and M. Kapalka. On the correctness of transactional memory. In *Principles and Practice of Parallel Programming*, PPOPP '08. ACM, 2008.
- [22] P. Hammarlund, R. Kumar, R. B. Osborne, R. Rajwar, R. Singhal, R. D'Sa, R. Chappell, S. Kaushik, S. Chennupati, S. Jourdan, et al. Haswell: The fourth-generation Intel core processor. *IEEE Micro*, (2), 2014.
- [23] M. Herlihy and J. E. B. Moss. Transactional memory: Architectural support for lock-free data structures. In *International Symposium on Computer Architecture*, ISCA '93. ACM, 1993.
- [24] G. J. Holzmann. The model checker SPIN. *IEEE Transactions on Software Engineering*, 23(5), May 1997.
- [25] J. Huang. Stateless model checking concurrent programs with maximal causality reduction. In *Programming Language Design and Implementation*, PLDI 2015. ACM, 2015.
- [26] Intel. Hardware lock elision overview. <https://software.intel.com/en-us/node/683688>, 2013.
- [27] C. S. Jensen, A. Møller, V. Raychev, D. Dimitrov, and M. Vechev. Stateless model checking of event-driven applications. In *Object-Oriented Programming, Systems, Languages, and Applications*, OOPSLA 2015. ACM, 2015.
- [28] B. Kasikci, C. Zamfir, and G. Candea. Data races vs. data race bugs: Telling the difference with portend. In *Architectural Support for Programming Languages and Operating Systems*, ASPLOS XVII. ACM, 2012.
- [29] K. P. Lawton. Bochs: A portable PC emulator for Unix/X. *Linux J.*, 1996(29es), Sept. 1996.
- [30] T. Leesatapornwongsa, M. Hao, P. Joshi, J. F. Lukman, and H. S. Gunawi. SAMC: Semantic-aware model checking for fast discovery of deep bugs in cloud systems. In *Operating Systems Design and Implementation*, OSDI'14. USENIX Association, 2014.
- [31] A. Mazurkiewicz. Trace theory. In *Advances in Petri Nets 1986, Part II on Petri Nets: Applications and Relationships to Other Models of Concurrency*. Springer-Verlag, 1987.
- [32] M. Musuvathi, S. Qadeer, T. Ball, G. Basler, P. A. Nainar, and I. Neamtiu. Finding and reproducing heisenbugs in concurrent programs. In *Operating Systems Design and Implementation*, OSDI'08. USENIX Association, 2008.
- [33] R. O'Callahan and J.-D. Choi. Hybrid dynamic data race detection. In *Principles and Practice of Parallel Programming*, PPOPP '03. ACM, 2003.
- [34] J. O'Leary, B. Saha, and M. R. Tuttle. Model checking transactional memory with Spin. In *Principles of Distributed Computing*, PODC '08. ACM, 2008.
- [35] E. Pozniansky and A. Schuster. Efficient on-the-fly data race detection in multithreaded C++ programs. In *Principles and Practice of Parallel Programming*, PPOPP '03. ACM, 2003.
- [36] K. Serebryany and T. Iskhodzhanov. ThreadSanitizer: Data race detection in practice. In *Workshop on Binary Instrumentation and Applications*, WBIA '09. ACM, 2009.
- [37] J. Simsa. *Systematic and Scalable Testing of Concurrent Programs*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 2013.
- [38] J. Simsa, R. Bryant, and G. Gibson. dBug: Systematic evaluation of distributed systems. In *Systems Software Verification*, SSV'10. USENIX Association, 2010.
- [39] J. Simsa, R. Bryant, and G. Gibson. Runtime estimation and resource allocation for concurrency testing. Technical Report CMU-PDL-12-113, Carnegie Mellon University, December 2012.
- [40] T. VII. What, if anything, is epsilon? In *Conference in Celebration of Harry Q. Bovik's 2⁶th Birthday*, SIGBOVIK '14. ACH, 2014.
- [41] T. VII. ZM~~ # PRinty# C with ABC! In *Conference in Celebration of Harry Q. Bovik's 2⁶th Birthday*, SIGBOVIK '17. ACH, 2017.
- [42] J. Yang, T. Chen, M. Wu, Z. Xu, X. Liu, H. Lin, M. Yang, F. Long, L. Zhang, and L. Zhou. MODIST: transparent model checking of unmodified distributed systems. In *Networked Systems Design and Implementation*, NSDI'09. USENIX Association, 2009.
- [43] Y. Yang, X. Chen, G. Gopalakrishnan, and R. M. Kirby. Efficient stateful dynamic partial order reduction. In *Workshop on Model Checking Software*, SPIN '08. Springer-Verlag, 2008.
- [44] R. M. Yoo, C. J. Hughes, K. Lai, and R. Rajwar. Performance evaluation of Intel(R) transactional synchronization extensions for high-performance computing. In *High Performance Computing, Networking, Storage and Analysis (SC)*. IEEE, 2013.
- [45] N. Zhang, M. Kusano, and C. Wang. Dynamic partial order reduction for relaxed memory models. In *Programming Language Design and Implementation*, PLDI 2015. ACM, 2015.



CONFIDENTIAL COMMITTEE MATERIALS

SIGBOVIK 2018 Paper Review

Paper 13: Transactional Memory
Concurrency Verification with Landslide

Assistant Professor

Rating: Thesis-Worthy

Confidence: External Committee Member

This paper presents earth-shaking work that deserves to be read by the entire computer science community. To that end, have you considered putting it in a thesis?



SIGBOVIK 2018

(Continued) Message from the Organizing Committee

You empty Waste Bay #1, but decide that you can wait until you get home to empty Waste Bay #2. Not wanting to risk encountering another dangerously conductive beverage, you exit the building.

goto PAGE_17;



SIGBOVIK 2018

(Continued) Message from the Organizing Committee

With mechanical precision, you easily repair your pants using a single thread. Multithreaded pant repair must only be necessary for humans, who could never stitch so precisely.

As you put your pants back on, you notice that you are surrounded by curious observers. Some look shocked that you so cavalierly removed your pants, while others look impressed at your pant repair prowess. Just in case any of the onlookers are Serious Researchers, you scurry outside before any of them realize you are a robot.

goto PAGE_17;

Save Me

22 **Dead programming**

Michael Coblenz

Keywords: programming languages, live programming,
dead programming

23 **Alternary operators: Alternative ternary operators**

Jim McCann; Your Name Here; and Your Name Here, Evan

Keywords: ternary, punctuation, operator, punchline

24 **bashcc: Multi-prompt one-shot delimited continuations
for Bash**

Spencer Baugh and Dougal Pugson

Keywords: control effects, delimited continuations, shell
scripting

25 **Towards a formalization of Claude Shannon's masters thesis**

andrewtron3000

Keywords: Coq, proof assistant, formalization, proposi-
tional logic, electromechanical relays

Dead Programming

Michael Coblenz
Carnegie Mellon University
Pittsburgh, Pennsylvania
mcoblenz@cs.cmu.edu

ABSTRACT

Live programming has recently become a popular topic of research. This paper introduces dead programming, a promising new direction in programming language design.

KEYWORDS

Programming languages, Live programming, Dead programming

ACM Reference Format:

Michael Coblenz. 2018. Dead Programming. In *Proceedings of ACH SIGBOVIK (SIGBOVIK '18)*. ACM, New York, NY, USA, 2 pages. <https://doi.org/10.1145/nnnnnnn.nnnnnnn>

1 INTRODUCTION

Live programming [1] espouses a responsive model in which the programming environment provides continuous feedback to the programmer regarding program behavior. Unfortunately, this approach to programming suffers from several flaws, which we address in our new design. First, in spite of a programmer's best effort, programs in live programming languages can include bugs; fixing these bugs in a live program is a hazardous occupation [2]. Second, as with traditional programming languages, programs written in live programs can have poor performance; consider the fact that live programs run on limited hardware and are written with traditional features such as recursion and loops. Third, live programs can be very expensive to construct, since they require expensive programmers to write them.

Dead programming addresses all of the above problems with a novel programming model. In fact, dead programming consists of a whole suite of tools, which enable superior programmer productivity. In this paper, we describe the first programming language, DEAD, designed to support dead programming, along with a compiler and optimizer. Although DEAD programs can be executed on traditional hardware, we also show how novel hardware can optimize execution beyond that which currently-available production equipment can attain.

2 PROGRAMMING LANGUAGE

In this section, we describe the programming language, DEAD, which we have successfully used to write dead software.

2.1 Syntax

$P ::= .$
| $P \text{ NOP}$

$V ::= ()$

2.2 Static Semantics

DEAD only supports one value, $()$, to which every program evaluates. Every program has type unit .

$$\frac{}{\vdash \cdot : \text{unit}} \qquad \frac{\vdash P : \text{unit}}{\vdash P \text{ NOP} : \text{unit}}$$

2.3 Dynamic semantics

Every DEAD program evaluates to the value $()$. The proof is left to the reader as a simple exercise in induction on the dynamic semantics.

$$\frac{}{\vdash \cdot \Downarrow ()} \qquad \frac{\vdash P \Downarrow v}{\vdash P \text{ NOP} \Downarrow ()}$$

The astute reader may notice that programs written in DEAD have no side effects, since the semantics require no operations on a store. DEAD is a pure functional language, as are most new, trendy languages. As is widely understood, mutable state is a primary cause of difficulty reasoning about program behavior. DEAD addresses this problem in a similar fashion to Haskell, another widely-used pure functional language. We expect that by facilitating ease of understanding and being substantially simpler than Haskell, DEAD will quickly become a very popular language, as it is a particularly convenient way of expressing programs that do nothing.

Some readers may be troubled by the lack of expressivity of a program that has no side effects. We remind the reader that, in the end, the output of every computer is a matter of converting electricity to heat, an operation that can be conducted quite effectively by a DEAD program.

2.4 Runtime environment

Conveniently, most modern CPUs already implement DEAD directly because they support a NOP instruction. Thus, the implementation of a runtime environment is quite straightforward and requires no overhead, unlike most existing approaches to programming. The use of a compiler is quite unnecessary. However, many programmers will avail themselves of an optimizer.

2.5 Compiler optimization

We show in this section how the traditional liveness analysis can be applied to DEAD programs. Note that no DEAD instruction reads from any register or memory address, which means that none of the temps written to are live. Thus, a correctly-implemented optimizer can remove every instruction from a DEAD program, leaving an optimal (zero-length) program.

As there are no facilities to invoke external library functions, dead code removal can safely remove all existing library code.

SIGBOVIK '18, April 1, 2018, Pittsburgh, PA USA
2016. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...\$15.00
<https://doi.org/10.1145/nnnnnnn.nnnnnnn>

The result is that, although DEAD is an expressive language, supporting programs of arbitrary size, every DEAD program optimizes optimally down to a zero-length program. Thus, the programmer can choose exactly how much electricity to convert to heat when running the program for a given execution environment, thus exactly replicating the behavior of a program written in a traditional programming language.

2.6 Hardware support

Current hardware environments offer efficient conversion of electricity to heat. However, in many cases this side effect is undesirable. We propose, then, a new programming environment, which is better-suited to running DEAD programs. We have successfully executed DEAD programs on a Russet potato. In addition to being an aesthetically pleasing addition to any office environment, the POTATO machine supports local farmers. It does have the disadvantage of eventually sprouting non-dead appendages; this is an opportunity for future work.

3 THE FUTURE: UNDEAD PROGRAMMING

Undead programming represents the next frontier in programming environments. Although one might argue that the POTATO execution environment is already undead, future work should more thoroughly explore the space of undead languages and environments.

REFERENCES

- [1] Sean McDirmid. 2007. Living It Up with a Live Programming Language. In *Proceedings of the 22nd Annual ACM SIGPLAN Conference on Object-oriented Programming Systems and Applications (OOPSLA '07)*. ACM, New York, NY, USA, 623–638. <https://doi.org/10.1145/1297027.1297073>
- [2] Aplana Software. 2015. When developers try to fix a bug in production. <https://www.youtube.com/watch?v=75wa8Lx4yc4>. (2015).

Alternary Operators: Alternative Ternary Operators

Jim McCann*
TCHOW llc

Your Name Here

Your Name Here, Evan

A ? B : C

Figure 1: *The ternary traditional operator (“tradnary operator”) evaluates to either B (if A evaluates to true) or C (otherwise).*

Abstract

Wizened old C++ programmers often refer reverently to “the” ternary operator – a wonderful language construct that allows in-line decision-making to compactly expressed.

But is the traditional ternary operator really *the* ternary operator, or are there other, as-yet-undiscovered, ternary operators that work similarly? In this paper, we axiomatize the notion of a ternary operators and explore the space of operators consistent with these axioms, demonstrating that there is, indeed, no one true ternary operator.

CR Categories: [Blank]: Blankings—Blank

1 Introduction

The traditional ternary operator (“tradnary operator”) is used in C-like languages [ISO 2017][†] to express the notion of in-line decision making:

$$A ? B : C$$

Where *B* and *C* must have compatible types. At runtime, first *A* is evaluated, and then either *B* or *C* are evaluated based on the truth-value of *A* (particularly, if *A* is true, then *B* is evaluated; otherwise *C* is evaluated). See also Figure 1.

This notion of fickle evaluation gives the tradnary operator a lot of power, on both sides of the equals sign.

For instance, one can use it to implement circular indexing:

```
next = (i < size ? arr[i] : arr[0]);
```

Provide a default value for missing data:

```
val = (pts[i] ? *pts[i] : 0.0f);
```

Pick a variable to increment:

```
(x > 0 ? pos_sum : neg_sum) += x;
```

Pick a vector to append to:

```
(x > 0 ? pos : neg).push(x);
```

Or even select a function to call:

*e-mail: ix@tchow.com

[†]Though the word “ternary” doesn’t actually appear in the specification. ISO/IEC calls it the conditional operator.

```
(op == '+' ? add : sub)(a, b);
```

2 Which Operators are Ternary Operators?

What is it that makes the tradnary operator a member of the fraternity of ternary operators (“fraternary”)? We believe that it must hew to several key ternary properties (“terperities”). These properties are largely non-surprising, and so are stated with minimal justification.

2.1 List of Fraternary Terperities

Aritude. To be ternary, an operator must have three operands. So while it might be tempting to define an operator of the form $A ? B : C : D$, this would not fit our definition. As they say in the culture: “to be a cool dude, you gotta have the right aritude.”

Infixulation. Like all C operators, ternary operators should be written in infix notation using characters that are disallowed in variable names. Ideally, the punctuation should be such that a lexer can tokenize the operator without resorting to recursion or complicated pattern matching. This may seem like a limitation in the creation of additional operators, but – even though the current specification seems to recommend it – in practice, most compilers[‡] reject non-ASCII letters in variable names, leaving whole swaths of codepoints available for productive use, including `!`, `¿`, and `U+203D`.

Indeed, operator overloading also allows the re-use of the same punctuation as the tradnary operator; though this concept does not persist throughout all the languages that support the operator, so alternate punctuation may be preferred.

Fickleness. Of course, what makes any ternary truly interesting is its mercurial nature. The fact that the operator evaluates only some of its operands, in a way that cannot – in general – be determined at compile time, is integral to its nature.

Uniquivity. Finally, it should be the case that any ternary operator would be cumbersome to duplicate with other language syntax. Impossibility is too much to ask, since – as we

[‡]As in, both of the ones we tried.

show later – even the tradnary operator can be replaced with a templated functor.

3 Alternative Ternary Operators

With these properties in hand, we can finally consider what alternative ternary operators (“alternary operators”) may exist that are Fickle, Infixulated, Uniqued, and have the right Aritude. Below, we review some possible operators and include implementation notes.

3.1 The Race Condition

$$A \# B \text{ ☒ } C$$

This ternary operator evaluates A and B in parallel, and takes the value of whichever one finishes fastest. The expression C is evaluated only in cases where A completes before B.

3.2 The Reverse Tradnary

$$A : B ? C$$

If evaluating C after A would produce a true value, then evaluate A, otherwise evaluate B.

NOTE: a general implementation likely requires transactional memory and/or use of observable speculative execution (e.g. the spectre “bug”); while limiting the operator to compile-time evaluation only would, of course, violate *fickleness*. A practical middle-ground would be to require A to be an expression for which the compiler can statically determine all referenced variables.

3.3 The Iternary

$$A @ B : C$$

If B is true, evaluate A while B remains true. The value of the expression is the last value of A. Otherwise, the value of the expression is the value of C.

3.4 The Internary

$$A \text{ 🙄 } B : C$$

The Internary is a cheaper Tradnary operator that only functions for a period of up to six months, negotiated at compile time and subject the to the availability of the Internary. Recompiling the code requires the Internary’s contract to be re-negotiated. Internarys may return as a Tradnary operator after completing their Tradnary schooling.

3.5 The Caternary

$$A \text{ 🐾 } B : C$$

The result of A is used as a hash to select containers of different sizes and materials, labeled B and C. A cat is placed

in a room with containers B and C. When a cat sits in a container, the corresponding expression is evaluated. The initial cat may choose to bring other cats into the room (“concatination”), in which case expressions B and C may be evaluated multiple times at non-deterministic intervals. This operator may deadlock due to lack of interest from the selected cats.

3.6 Schrödinger’s Caternary

$$A \Psi B : C$$

This is similar to the Caternary operator except that there is only one container in a room labeled both B and C and the room is set up as a Schrödinger’s Cat experiment. After a random interval determined by evaluating A, if the cat is found inside the box and is alive, B is evaluated. If the cat is found inside the box and is dead, C is evaluated. If the cat is found outside the box chasing a laser instead, the experiment is reset.

3.7 The Facebook™ Sponsored Operators

Thanks to a generous and entirely fictitious[§] grant from Facebook™, we have been able to posit a set of alternary operators with deep social media integration:

$$\begin{aligned} A \text{ 👍 } B : C \\ A \text{ ❤️ } B : C \\ A \text{ 😊 } B : C \\ A \text{ 😱 } B : C \\ A \text{ 😞 } B : C \\ A \text{ 🙄 } B : C \end{aligned}$$

These operators post the result of evaluating A to your Facebook™ feed and – after enough time has elapsed – check if the predominant reaction matches the one used in the operator. If so, the value of the expression is the result of evaluating B; if not, the value of the expression is the result of evaluating C.

NOTE: Lack of reactions may lead to deadlock. Overuse of these operators may lead to a lack of friends, followed by deadlock.

3.8 The Hardest Ternary Operator Ever

A tip o’ the hat to George Boolos for inspiring this difficult operator:

$$A \square B \diamond C$$

Your operands will be evaluated by three gods, named True, False, and Random. You do not know which statement will be evaluated by which god, or in which order. True will evaluate the statement and take the result directly, False will evaluate the statement and take the result directly, False will evaluate the statement and return the opposite of the result,

[§]The generosity is, of course, real.

Random does not evaluate the statement and returns a random result. The overall value of the expression will be either the string “da” or “ja”, but the mapping between the strings and the values `true` and `false` is not known.

3.9 Evan’s Prerequisite Dragon-Related Operator

A \$ B 🐉 C

The result of evaluating `A` is presented to a dragon[¶] as tribute. If the dragon accepts your offering, it adds `A` to its hoard and evaluates `B`. Otherwise, your offering is deemed unacceptable and the dragon begins a rampage of burnination over the countryside while evaluating `C`.

4 General Ternary Operators

The ternary operators discussed in the previous section remain purely theoretical. However, the astute C++ programmers among you will have realized that all of the operators we have discussed conform to a relatively simple functor signature^{||}:

```
template< char C1, char C2, typename TA,
          typename TB, typename TC >
struct ternary;
```

Where specializations define an operator():

```
template< typename TR >
TR operator() (
    std::function< TA() > const &A,
    std::function< TB() > const &B,
    std::function< TC() > const &C
);
```

With the appropriate specializations defined, the compiler can simply “de-sugar” the expression

A ? B : C

into

```
ternary< '?', ':', TA, TB, TC >() (
    [&]()->TA{A}, [&]()->TB{B}, [&]()->TB{C})
```

So, for instance, the ternary operator can be supported by providing the following specialization^{**}:

```
template< typename TA, typename TR >
struct ternary<'?', ':', TA, TR, TR > {
    TR operator() (
        std::function< TA() > const &A,
        std::function< TR() > const &B,
```

[¶]Don’t have a dragon handy? Just e-mail dragon@cmu.edu.

^{||}At least, as soon as memory transactions make it into the standard library.

^{**}This does gloss over some type coercion that the actual ternary operator performs. But we’re about 80% sure that one could express this with sufficient template wrangling.

```
std::function< TR() > const &C
) {
    if (A()) return B();
    else return C();
}
};
```

With, e.g., our second initial example de-sugaring as:

```
// (op == '+' ? add : sub)(a,b);
ternary< '?', ':', bool,
int(*) (int,int), int(*) (int,int) >() (
    [&]()->bool {return op == '+';},
    [&]()->int(*) (int,int) {return add;},
    [&]()->int(*) (int,int) {return sub;}
)(a,b);
```

Which is definitely enough to make one really, really appreciate syntactic sugar.

5 Future Work

- Quaternary Operators
- Veterinary Operators
- Veteranary Operators

6 Conclusion and/or Punchline

In this paper, we demonstrated that by axiomatizing “the” ternary operator – that is, extracting a list of ternary properties describing it – a whole slew of new ternary operators could be defined. Like all good programming language constructs, these operators range from the practical to the absurd.

In addition, thanks to the power of modern C++ templating, we showed how at least one language could add support for new first-class ternary operators through desugaring. Though it’s worth noting that we glossed over the question of how a compiler is supposed to tokenize punctuation marks when it doesn’t know which marks are valid until after parsing a file. Command line options, I guess?

Regardless, it isn’t “the” ternary operator after all; it is simply one among many possibilities. So, in the end, we find that wizened old C programmers have been wrong all along.

References

ISO. 2017. *International Standard ISO/IEC 14882:2017(E) Programming Language C++*. International Organization for Standardization, Geneva, Switzerland, Mar. I haven’t actually read this and I probably should but you know how the time gets away from you.

bashcc: Multi-prompt one-shot delimited continuations for bash

Spencer Baugh
University of Carcosa
104 Lost Beach
Carcaso, Hyades
first.last@gmail.com

Dougal Pugson
Pugson's C++ Crypt LLC
New York, USA
dougalpugson@gmail.com

Abstract

Among functional programmers, continuations are well known for the influence they have on the simplicity and understandability of a program. And among sysadmins, the bash programming language is renowned for the maintainability of programs written in it. Unfortunately, until this point, shell programmers have been denied the ability to use continuations in their programs. We provide an implementation of delimited continuations in GNU bash. This will provide a more familiar programming environment for functional programmers writing bash, and give bash programmers access to the advanced abstraction techniques of modern functional languages. We provide implementations of exceptions, early return, and coroutines as motivation, and outline areas for future work.

CCS Concepts •Software and its engineering →Control structures; Scripting languages; •Social and professional topics →Offshoring;

Keywords SIGBOVIK, effect handler, delimited continuations, bash, shells

ACM Reference format:

Spencer Baugh and Dougal Pugson. 2018. bashcc: Multi-prompt one-shot delimited continuations for bash. In *Proceedings of SIGBOVIK, Pittsburgh, PA USA, April 2018 (SIGBOVIK 2018)*, 4 pages. DOI: 10.475/123.4

1 Introduction

From the beginning¹, there has been a close relationship between the Unix shell and functional programming. The popular functional programming technique “currying” was first developed as part of the BSD Unix shell in 1983, as an implementation detail of the “rsh” monad. Indeed, the term “function” itself comes from the notation used to begin a subroutine in a Unix shell script, `function`.

As time has gone on, these two communities have grown in separate directions. The functional programming community and shell scripting community have each developed their own advanced techniques, with little cross-pollination.

We sought to rectify this by porting a simple technique from functional programming languages to GNU bash. Delimited continuations are a straightforward mechanism, useful for implementing dynamic variables, going backwards in time, and in a pinch, providing a healthy meal of *continuatí al pomodoro*.

¹the point at which our universe’s continuation is delimited by the Prime Reset

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

SIGBOVIK 2018, Pittsburgh, PA USA

© 2018 Copyright held by the owner/author(s). 123-4567-24-567/08/06...\$15.00

DOI: 10.475/123.4

Unix shells have long had “shift”, one half of what is necessary for delimited continuations. Unfortunately, Ken Thompson apparently didn’t read Oleg Kiselyov’s website closely enough, so he didn’t bother to implement “reset” as well.

Rather than build our continuation library on top of the existing “shift” infrastructure, we thought it was best to get a clean start. Hence we implemented “run” and “yield”, following the naming convention of [1], but extending to work with multiple prompts.

For more background on delimited continuations, consult <http://okmij.org/ftp/continuations/#tutorial> or [2].

Our paper is organized in some number of sections. Section 1 contains an introduction and a description of the organization of the paper. Section 2 demonstrates several applications. Section 3 gives an overview of the implementation of delimited continuations in bash. Section 4 concludes the paper, discusses future work, and affirms that this was the right thing to do.

2 Applications

Our API is based on “run” and “yield”, and “invoke” to resume continuations. For more details on the API, see 3.

2.1 Exceptions

We have implemented exceptions in bash using delimited continuations. The implementation of `raise` is simplicity itself:

```
function raise () {
    yield $exception_handler "$@"
}
```

We can raise whatever string we want as an exception by yielding it to the handler. `try` is implemented as follows.

```
function try () {
    local exception_handler=$(make_prompt)
    response=$(run $exception_handler "$@" )
    while ;;
    do
        if [[ $response =~ $yield_re ]]
        then
            continuation=${BASH_REMATCH[1]}
            exception=${BASH_REMATCH[2]}
            return 1
        elif [[ $response =~ $return_re ]]
        then
            file=${BASH_REMATCH[1]}
            cat $file
            return 0
        fi
    done
}
```

Beautifully simple! We run an expression under a prompt, and case on the possible responses. If the expression returns without throwing, we propagate the result up (with `cat`). If the expression does throw, we'll see that as a `yield`, and we set "exception" and return non-zero.

Note that to achieve the dynamic scoping behavior of exception handlers, this uses bash's "local" keyword, which provides dynamic scope in bash.

Then usage is as simple as this:

```
despicable=12
function liker () {
    if [[ $1 -eq $despicable ]];
    then raise "I_\don't_\like_\$1!!!"
    else echo "I_\like_\$1" >&2
    fi
}
function evaluator () {
    seq 20 | while read number;
    do liker $number;
    done
}
function main () {
    try evaluator || {
        echo "exception_\was_\$exception"
        exit 1
    }
}
```

This will print numbers, until it gets to a despicable number (12), and then throw an exception and abort. Even such practical functionality as this is improved by delimited continuations!

2.2 Early return

Dealing with exceptions is hard and annoying, so we have implemented an "early return" functionality, which allows a function to immediately return a value at the enclosing prompt, without requiring a wrapper function or forcing the user to catch an exception.

Consider a simple recursive function which multiplies all its arguments:

```
function multiply_args () {
    first=$1; shift
    if [[ $# -eq 0 ]]; then
        echo $first
    else
        rest=$(multiply_args "$@")
        echo $(( first * rest ))
    fi
}
multiply_args 1 2 0 4 5
```

As expected, this prints 0.

However, this function is very inefficient! It keeps iterating over its arguments even if it sees 0! We can improve it with continuations. We add this `early_return` functionality.

```
function early_return () {
    prompt=$1; shift
```

```
    file=$(name early_return .XXXX)
    echo "$@" > $file
    file_send_line $prompt "return:_$file"
    exit 0
}
```

Then we can change our function to call `early_return` if it sees a 0.

```
function multiply_args () {
    prompt=$1; shift
    first=$1; shift
    if [[ $# -eq 0 ]]; then
        echo $first
    elif [[ $first -eq 0 ]]; then
        early_return $prompt 0
    else
        rest=$(multiply_args $prompt "$@")
        echo $(( first * rest ))
    fi
} # $
prompt=$(make_prompt)
run_with_prompt $prompt multiply_args \
    $prompt 1 2 0 4 5
```

Now it will be much more efficient, because it will only iterate down to the first 0 in the argument list, and then early return to print 0 immediately!

We omit the straightforward implementation of `run_with_prompt` as to not overly inflate our page count.

2.3 Coroutines

Coroutines are a recent programming technique invented by the "Golang" programming language. It allows several concurrent sequential processes to communicate by sending messages over a channel. While this sounds useless, it does have a few niche applications, and so we have implemented it in bash. As is traditional for coroutine implementations, we have not implemented actual parallel execution for coroutines.

Due to censorship by the squeamish weaklings on the SIGBOVIK review committee, we were not able to include our coroutine implementation in this paper.

But we wish to assure you that coroutines definitely work with this framework. The traditional primitives of "spawn", "send" and "recv" are all there.

We have received queries about whether we used the bitwise-or operator, `|`, in our coroutine implementation. While we are confused why anyone would ask such a thing, we would like to assure you that `|` is not used for any of the core functionality in our bash coroutine implementation.

Please view our implementation on Github for more: <https://github.com/catern/bashcc>.

3 Implementation

The implementation is guided by the insight of [3], which implemented multi-prompt one-shot delimited continuations using threads and synchronization primitives.

It occurred to the author that such thread-based continuations could be made multi-shot trivially, by simply switching the implementation to use processes and message passing instead, and then when invoking a continuation (by passing a message) using fork on the receive side to duplicate the continuation.

We discovered that continuations and processes have the follow correspondence:

Continuations	Processes
shift	blocking message send
reset	blocking message receive
invoking a continuation	replying to message

Understanding and formalizing this correspondence is left as an exercise for the reader.

Unfortunately, bash doesn't actually support fork, that is, returning twice. It only supports spawning new processes. This constrains us to implementing only one-shot continuations. As bash is a untyped language, supportingly only one type, we felt it was thematically appropriate that its continuations be uncontinuations, supporting only one invocation.

Also, since bash does not have message passing functionality natively anyway, and it's a major pain to deal with pipes in bash, we implemented our message passing by reading and writing to files.

First off, we define creating new prompts as creating a new empty file.

```
function make_prompt() {
    prompt=$(name prompt.XXXX)
    touch $prompt
    echo $prompt
} # $
```

We pass the path of the file whenever we need to refer to the prompt.

Next, we define "run", and its return value. "run" takes a prompt and a command and runs the command under the prompt. If the command yields, then "run" returns the yielded value, along with the continuation, tagged with "yield:". If the command returns, then "run" returns the return value, tagged with "return:".

If the command is going to yield, it needs to take the prompt as an argument explicitly, because "run" does not pass the prompt in.

```
function run() {
    prompt=$1; shift
    prompt_size=$(file_size $prompt)
    stdout=$(name stdout.XXXX)
    { "$@";
        file_send_line $prompt \
            "return:_$stdout";
    } >$stdout </dev/null &
    file_wait_for_one_line $prompt $prompt_size
}
```

"yield" takes an arbitrary line of data, sends it to the prompt (appropriately framed as a yield), then returns the line of data that the prompt replied with.

It's here that we create the continuation. The continuation, like the prompt, is a file. We send the filename of our continuation to the prompt along with the yield data. The prompt will write the resume data to our continuation file, and we'll resume.

```
function yield() {
    prompt=$1; shift
    continuation=$(name continuation.XXXX)
    touch $continuation
    file_send_line $prompt \
        "yield:_$continuation_message:" "$@"
    file_wait_for_one_line $continuation 0
} # $
```

Finally, we need to be able to invoke a continuation. "invoke" takes a prompt, a continuation, and an arbitrary line of data, and resumes that continuation with that data under that prompt. It returns the next YieldValue, just like "run".

```
function invoke() {
    prompt=$1; shift
    continuation=$1; shift
    prompt_size=$(file_size $prompt)
    file_send_line $continuation "$@"
    file_wait_for_one_line $prompt $prompt_size
}
```

For more details on the implementation, we have made our bash code available in the form of a Github repository, <https://github.com/catern/bashcc>.

4 Conclusion

We find that delimited continuations in bash are a great improvement in expressive power, and allow us to implement coroutines, early return, exceptions, and state, all in native, bare-metal 100% pure uncut Colombian GNU bash.

4.1 Future work

4.1.1 Multi-shot continuations

The most obvious direction for future work is to support multi-shot continuations. The missing piece is the ability to fork in bash. There are several ways this could be achieved.

We could provide a new bash builtin which exposes "fork" as a primitive. Unfortunately, that would need to be compiled against the end-user's bash system, an unacceptable deployment problem.

Instead, we can use gdb to attach to bash, force it to execute a "fork" syscall, and then detach. This is easy to deploy since most systems already have gdb installed, so this approach has absolutely no issues at all.

4.1.2 Type-and-effect system

We could provide a type-and-effect system for bash. Since bash does not have a type system, this would actually just be an effect system, no types.

This could likely be implemented through an additional compilation phase of bash, using a notion called "name inference". In the tradition of Scheme ending effectful functions in "!", we would rename functions to include their effects. For example, a function `f` which uses exceptions and IO would be automatically renamed to `f_effects:_exceptions_io`. Then it would be impossible for a programmer to use a function without knowing its effects, as it should be.

4.2 Conclusion conclusion

We conclude our conclusion with the hope that many new bash programs are written using these features. With these features, other languages such as Python and OCaml are permanently obsoleted, and bash is triumphant. We expect that the authors are likely to be remembered forever by the maintenance programmers of the future.

References

- [1] Roshan P. James and Amr Sabry. 1983. Yield: Mainstream Delimited Continuations.
- [2] Oleg Kiselyov. 2012. Delimited control in OCaml, abstractly and concretely. *Theoretical Computer Science* 435 (2012), 56 – 76. DOI: <http://dx.doi.org/https://doi.org/10.1016/j.tcs.2012.02.025> Functional and Logic Programming.
- [3] Sanjeev Kumar, Carl Bruggeman, and R. Kent Dybvig. 1998. Threads Yield Continuations. *LISP and Symbolic Computation* 10, 3 (01 May 1998), 223–236. DOI: <http://dx.doi.org/10.1023/A:1007782300874>

Towards a Formalization of Claude Shannon’s Masters Thesis

andrewtron3000*

March 8, 2018

1 Overview

Claude Shannon’s seminal MS thesis [3] is considered by many to be the most important masters thesis of the 20th century.

Shannon’s thesis laid the foundation for the digital revolution by showing how to reason about electromechanical relay circuits using boolean algebra and propositional logic.

Shannon’s thesis contains many theorems and assertions – most of which are not proven. This paper sets out to prove many of those theorems and assertions.

Interested readers can obtain a copy of Shannon’s M.S. thesis in the link provided in the reference. Most of the theorems are numbered in the original thesis and this paper refers to those numbers. Occasionally an assertion is not uniquely identifiable in the thesis, and in this case, a page number is used to identify it.

The available copy of the thesis is a scanned PDF of a typewritten manuscript. The quality of the scan is somewhat poor and this was another motivation to subject the contents of the thesis to more rigorous examination. Despite being published in 1936, we find that the typography is sorely lacking: no doubt a harbinger of the degradation of typography during the 20th century [2].

*<https://github.com/andrewtron3000/shannon-coq>

2 Postulates

In order to define the circuit algebra, we first introduce the notion of a circuit, which can be either closed (conducting) or open (non-conducting). This is postulate 4 from the thesis. We specify this postulate as an inductive type in Coq:

```
Inductive circuit : Type :=  
| closed : circuit  
| open : circuit.
```

Next, we define the notion of “plus” in the circuit algebra. Plus is synonymous with two circuits in series, or the AND operation in boolean algebra. This is defined in postulates 1b, 2a and 3a. We use a Coq definition to formalize this notion.

```
Definition plus (v1 v2 : circuit) : circuit :=  
  match v1, v2 with  
  | open, open ⇒ open  
  | open, closed ⇒ open  
  | closed, open ⇒ open  
  | closed, closed ⇒ closed  
  end.
```

Next, we define the notion of “times” in the circuit algebra. Times is synonymous with two circuits in parallel (or the OR operation in boolean algebra) and it is defined in postulates 1a, 2b, and 3b.

```
Definition times (v1 v2 : circuit) : circuit :=  
  match v1, v2 with  
  | closed, closed ⇒ closed  
  | open, closed ⇒ closed
```

```
| closed, open ⇒ closed
| open, open ⇒ open
end.
```

Now we define some convenient Coq notation for these plus and times functions so that further developments can use the normal symbols for plus and times.

```
Notation "x + y" :=
  (plus x y)
  (at level 50,
   left associativity).
```

```
Notation "x * y" :=
  (times x y)
  (at level 40,
   left associativity).
```

Next we will introduce the definition of negation. As one would expect, negation of an open circuit is closed and negation of a closed circuit is open.

```
Definition negation (v1 : circuit) : circuit :=
  match v1 with
  | open ⇒ closed
  | closed ⇒ open
  end.
```

3 Proofs of First Theorems

With these postulates defined, we can begin using them to prove theorems 1 through 5.

Throughout this development, with just a few exceptions, our goal is to have all proofs proven with a single tactic application, following the Chlipala discipline [1].

To do this, we start by defining a set of custom Ltac tactics below.

```
Ltac reduce1 X :=
  try destruct X;
  simpl;
  reflexivity.
```

```
Ltac reduce2 X Y :=
  try destruct X;
  try destruct Y;
  simpl;
```

```
  reflexivity.
```

```
Ltac reduce3 X Y Z :=
  try destruct X;
  try destruct Y;
  try destruct Z;
  simpl;
  reflexivity.
```

Now we state Theorem 1a (plus over circuits is commutative) and prove it in a straightforward fashion.

```
Theorem plus_comm : ∀ (x y : circuit),
  x + y = y + x.
```

Proof.

```
  intros X Y.
  reduce2 X Y.
```

Qed.

Next we state Theorem 1b (times over circuits is commutative) and prove it.

```
Theorem times_comm : ∀ (x y : circuit),
  x * y = y * x.
```

Proof.

```
  intros X Y.
  reduce2 X Y.
```

Qed.

Next we prove Theorem 2a – that plus is associative.

```
Theorem plus_assoc : ∀ (x y z : circuit),
  x + (y + z) = (x + y) + z.
```

Proof.

```
  intros X Y Z.
  reduce3 X Y Z.
```

Qed.

Next we prove Theorem 2b – that times is also associative.

```
Theorem times_assoc : ∀ (x y z : circuit),
  x * (y * z) = (x * y) * z.
```

Proof.

```
  intros X Y Z.
  reduce3 X Y Z.
```

Qed.

Next, we prove Theorem 3a – that times is distributive.

Theorem `times_dist` : $\forall (x\ y\ z : \mathbf{circuit}),$
 $x * (y + z) = (x * y) + (x * z).$

Proof.

```
intros X Y Z.
reduce3 X Y Z.
```

Qed.

Next, we prove Theorem 3b, that plus is also distributive.

Theorem `plus_dist` : $\forall (x\ y\ z : \mathbf{circuit}),$
 $x + (y * z) = (x + y) * (x + z).$

Proof.

```
intros X Y Z.
reduce3 X Y Z.
```

Qed.

Now we get to Theorem 4a which is a theorem about how times works when the first argument is the value open.

Theorem `open_times_x` : $\forall (x : \mathbf{circuit}),$
 $\text{open} * x = x.$

Proof.

```
intros X.
reduce1 X.
```

Qed.

And we prove Theorem 4b which is a theorem about how plus works when the first argument is the value closed.

Theorem `closed_plus_x` : $\forall (x : \mathbf{circuit}),$
 $\text{closed} + x = x.$

Proof.

```
intros X.
reduce1 X.
```

Qed.

Next, Theorem 5a asserts a relationship of plus when the first argument is the value open.

Theorem `open_plus_x` : $\forall (x : \mathbf{circuit}),$
 $\text{open} + x = \text{open}.$

Proof.

```
intros X.
reduce1 X.
```

Qed.

And Theorem 5b asserts a relationship of times when the first argument is closed.

Theorem `closed_times_x` : $\forall (x : \mathbf{circuit}),$

$\text{closed} * x = \text{closed}.$

Proof.

```
intros X.
reduce1 X.
```

Qed.

4 Negation Theorems

Theorem 6a asserts the behavior when a circuit and its negative are connected in series. As you might expect, this always results in an open circuit.

Theorem `plus_neg` : $\forall (x : \mathbf{circuit}),$
 $x + (\text{negation } x) = \text{open}.$

Proof.

```
intros X.
reduce1 X.
```

Qed.

And Theorem 6b specifies what happens when you connect a circuit and its negative in parallel. As you would expect, the circuit is always closed in this case.

Theorem `times_neg` : $\forall (x : \mathbf{circuit}),$
 $x * (\text{negation } x) = \text{closed}.$

Proof.

```
intros X.
reduce1 X.
```

Qed.

Theorems 7a and 7b specify what happens when you negate the specific values of open or closed. These are quite simple and are formalized below.

Theorem `closed_neg` :
 $\text{negation closed} = \text{open}.$

Proof.

```
reduce1 X.
```

Qed.

Theorem `open_neg` :
 $\text{negation open} = \text{closed}.$

Proof.

```
reduce1 X.
```

Qed.

Theorem 8 specifies what happens when you take the negative of the negative of a circuit. As expected one gets the original circuit back.

Theorem `double_neg` : $\forall (x : \text{circuit}),$
`negation (negation x) = x.`

Proof.

`intros X.`
`reduce1 X.`

Qed.

5 Equivalence to Calculus of Propositions

Claude then describes how the algebra defined above is equivalent to propositional logic. He does this by showing an equivalence between the algebra above and E.V. Huntington's formulation of symbolic logic. This formulation has 6 postulates and postulates 1, 2, 3, and 4 are clearly met without proof. Postulates 5 and 6 of E.V. Huntington's formulation are proved below.

Theorem `plus_same` : $\forall (x y : \text{circuit}),$
`x = y →`
`x + y = x.`

Proof.

`intros X Y.`
`intros H.`
`rewrite → H.`
`reduce1 Y.`

Qed.

Theorem `dist_neg` : $\forall (x y : \text{circuit}),$
`(x * y) + (x * (negation y)) = x.`

Proof.

`intros X Y.`
`reduce2 X Y.`

Qed.

We can, for completeness also prove the definition mentioned in proposition 6.

Theorem `dist_neg_defn` : $\forall (x y : \text{circuit}),$
`(x * y) = negation ((negation x) + (negation y)).`

Proof.

`intros X Y.`

`reduce2 X Y.`

Qed.

6 A Proof of De Morgans Law

Once this equivalence between the circuit algebra and propositional logic is shown, it is possible to bring over powerful theorems from propositional logic into our new algebra. We will begin by proving De Morgan's theorem. This is Theorem 9.

While the thesis asserts these theorems for an arbitrary number of variables, we will only illustrate proofs for two and three variables.

Theorem `demorgan_9a_2` : $\forall (x y : \text{circuit}),$
`negation (x + y) =`
`((negation x) * (negation y)).`

Proof.

`intros X Y.`
`reduce2 X Y.`

Qed.

Theorem `demorgan_9a_3` : $\forall (x y z : \text{circuit}),$
`negation (x + y + z) =`
`((negation x) * (negation y) * (negation z)).`

Proof.

`intros X Y Z.`
`reduce3 X Y Z.`

Qed.

And we will prove De Morgan's theorem over times over two and three variables. This is Theorem 9b.

Theorem `demorgan_9b_2` : $\forall (x y : \text{circuit}),$
`negation (x * y) =`
`((negation x) + (negation y)).`

Proof.

`intros X Y.`
`reduce2 X Y.`

Qed.

Theorem `demorgan_9b_3` : $\forall (x y z : \text{circuit}),$
`negation (x * y * z) =`

```

      ( (negation x) +
        (negation y) +
        (negation z) ).

```

Proof.

```

  intros X Y Z.
  reduce3 X Y Z.

```

Qed.

7 Onward to Taylor Series

Claude then starts the discussion of how to specify arbitrary functions in the circuit algebra. He starts by illustrating the capability to expand an arbitrary function into a Taylor series expansion.

In order to complete these proofs we introduce more Ltac tactic machinery. At this point it will be important for us to be able to leverage many of the above theorems in subsequent proofs. We encapsulate these theorems into a new set of Ltac tactics. The tactics are shown below.

```

Ltac wham :=
  try repeat ( (rewrite → closed_times_x;
               rewrite → closed_neg;
               rewrite → open_times_x) ||
              (rewrite → open_neg;
               rewrite → closed_times_x;
               rewrite → open_times_x) ||
              (rewrite → open_neg) ||
              (rewrite → closed_neg) ).

```

```

Ltac open_plus_bam :=
  try ( (rewrite → open_plus_x) ||
        (rewrite plus_comm;
         rewrite open_plus_x) ).

```

```

Ltac closed_plus_bam :=
  try ( (rewrite → closed_plus_x) ||
        (rewrite plus_comm;
         rewrite closed_plus_x) ).

```

```

Ltac open_times_bam :=
  try ( (rewrite → open_times_x) ||
        (rewrite times_comm;
         rewrite open_times_x) ).

```

```

Ltac closed_times_bam :=
  try ( (rewrite → closed_times_x) ||
        (rewrite times_comm;

```

```

        rewrite closed_times_x) ).

```

```

Ltac bam :=
  try repeat (closed_plus_bam ||
              open_plus_bam ||
              closed_times_bam ||
              open_times_bam).

```

```

Ltac wham_bam_1 X :=
  try (destruct X;
       wham; bam;
       reflexivity).

```

```

Ltac wham_bam_2 X Y :=
  try (destruct X, Y;
       wham; bam;
       reflexivity).

```

```

Ltac wham_bam_3 X Y Z :=
  try (destruct X, Y, Z;
       wham; bam;
       reflexivity).

```

```

Ltac wham_bam_4 W X Y Z :=
  try (destruct W, X, Y, Z;
       wham; bam;
       reflexivity).

```

```

Ltac wham_bam_5 V W X Y Z :=
  try (destruct V, W, X, Y, Z;
       wham; bam;
       reflexivity).

```

```

Ltac wham_bam_6 S V W X Y Z :=
  try (destruct S, V, W, X, Y, Z;
       wham; bam;
       reflexivity).

```

And now we have the appropriate machinery in place to be able to prove the Taylor series expansion on two variables shown in Theorems 10a and 10b, here called `taylorA` and `taylorB` respectively.

```

Theorem taylorA :
  ∀ (x y: circuit),
  ∀ (f : circuit → circuit → circuit),
  f x y =
    ( (x * (f open y)) +
      ((negation x) * (f closed y)) ).

```

Proof.

```

  intros X Y.

```

```

intros F.
wham_bam_2 X Y.
Qed.
Theorem taylorB :
  ∀ (x y: circuit),
  ∀ (f : circuit → circuit → circuit),
  f x y =
  ( ((f closed y) + x) *
    ((f open y) + (negation x)) ).

```

Proof.

```

intros X Y.
intros F.
wham_bam_2 X Y.

```

Qed.

We continue with the expansion of the Taylor series to the second variable as described in Theorems 11a and 11b.

```

Theorem taylor11a : ∀ (x y: circuit),
  ∀ (f : circuit → circuit → circuit),
  f x y =
  ( (x * y) *
    (f open open) ) +
  ( (x * (negation y)) *
    (f open closed) ) +
  ( ((negation x) * y) *
    (f closed open) ) +
  ( ((negation x) *
    (negation y)) *
    (f closed closed) ).

```

Proof.

```

intros X Y.
intros F.
wham_bam_2 X Y.

```

Qed.

```

Theorem taylor11b : ∀ (x y: circuit),
  ∀ (f : circuit → circuit → circuit),
  f x y =
  ( (x + y) +
    (f closed closed) ) *
  ( (x + (negation y)) +
    (f closed open) ) *
  ( ((negation x) + y) +
    (f open closed) ) *
  ( ((negation x) + (negation y)) +

```

```

(f open open) ).

```

Proof.

```

intros X Y.
intros F.
wham_bam_2 X Y.

```

Qed.

We skip the proofs of Theorem 12a and 12b as we have shown their validity in the two variable case above. We also leave the proof of Theorem 13 to future work.

At the end of the first paragraph on page 14, the thesis illustrates an example of finding the negative of a particular function using the generalization described in Theorem 13. We prove it here, but do not use the power of Theorem 13. Instead we can use our simple custom tactic with good results.

```

Theorem example1 : ∀ (w x y z: circuit),
  negation ( x +
    ( y *
      (z + w * (negation x)) ) ) =
  (negation x) *
  ( (negation y) +
    (negation z) *
    ((negation w) + x) ).

```

Proof.

```

intros W X Y Z.
wham_bam_4 W X Y Z.

```

Qed.

8 Simplification Theorems

Next Claude presents Theorems 14-18, useful for simplifying expressions.

```

Theorem x_plus_x_is_x : ∀ (x: circuit),
  x + x = x.

```

Proof.

```

intros X.
wham_bam_1 X.

```

Qed.

```

Theorem x_times_x_is_x : ∀ (x: circuit),
  x * x = x.

```

Proof.

```

intros X.

```

wham_bam_1 X.
Qed.

Theorem `x_plus_xy` : $\forall (x\ y: \mathbf{circuit}),$
 $(x + (x * y)) = x.$
Proof.
intros X Y.
wham_bam_2 X Y.
Qed.

Theorem `x_x_plus_y` : $\forall (x\ y: \mathbf{circuit}),$
 $x * (x + y) = x.$
Proof.
intros X Y.
wham_bam_2 X Y.
Qed.

Theorem `theorem16a` : $\forall (x\ y\ z: \mathbf{circuit}),$
 $(x * y) + (\mathbf{negation}\ x) * z =$
 $(x * y) + ((\mathbf{negation}\ x) * z) + (y * z).$
Proof.
intros X Y Z.
wham_bam_3 X Y Z.
Qed.

Theorem `theorem16b` : $\forall (x\ y\ z: \mathbf{circuit}),$
 $(x + y) * ((\mathbf{negation}\ x) + z) =$
 $(x + y) * ((\mathbf{negation}\ x) + z) * (y + z).$
Proof.
intros X Y Z.
wham_bam_3 X Y Z.
Qed.

Theorem `theorem17a` : $\forall (x: \mathbf{circuit}),$
 $\forall (f: \mathbf{circuit} \rightarrow \mathbf{circuit}),$
 $x * (f\ x) = x * (f\ \mathbf{open}).$
Proof.
intros X.
intros F.
wham_bam_1 X.
Qed.

Theorem `theorem17b` : $\forall (x: \mathbf{circuit}),$
 $\forall (f: \mathbf{circuit} \rightarrow \mathbf{circuit}),$
 $x + (f\ x) = x + (f\ \mathbf{closed}).$
Proof.
intros X.
intros F.
wham_bam_1 X.
Qed.

Theorem `theorem18a` : $\forall (x: \mathbf{circuit}),$
 $\forall (f: \mathbf{circuit} \rightarrow \mathbf{circuit}),$
 $(\mathbf{negation}\ x) * (f\ x) =$
 $(\mathbf{negation}\ x) * (f\ \mathbf{closed}).$

Proof.
intros X.
intros F.
wham_bam_1 X.

Qed.

Theorem `theorem18b` : $\forall (x: \mathbf{circuit}),$
 $\forall (f: \mathbf{circuit} \rightarrow \mathbf{circuit}),$
 $(\mathbf{negation}\ x) + (f\ x) =$
 $(\mathbf{negation}\ x) + (f\ \mathbf{open}).$

Proof.
intros X.
intros F.
wham_bam_1 X.

Qed.

9 Series Parallel Example

Figure 5 shows an example of an expression that represents a fairly complex series parallel circuit. The figure is first rendered into a hindrance equation. The equation is then manipulated into a simpler form. We prove that the transformation between Figure 5 and Figure 6 is correct.

Theorem `figure5` : $\forall (s\ v\ w\ x\ y\ z: \mathbf{circuit}),$
 $w + ((\mathbf{negation}\ w) * (x + y)) +$
 $(x + z) * (s + (\mathbf{negation}\ w) + z) *$
 $((\mathbf{negation}\ z) + y + (\mathbf{negation}\ s) * v) =$
 $w + x + y + z * (\mathbf{negation}\ s) * v.$

Proof.
intros S V W X Y Z.
wham_bam_6 S V W X Y Z.

Qed.

10 Multi-Terminal Networks

In this section, we discuss Section III of the thesis. Section III begins by illustrating two types of non-serial and non-parallel networks: the delta and wye circuit configurations.

We tackle the equivalence of Figure 8, the delta to wye transformation first. The path from a to b in the delta configuration is r in parallel with the both s and t in series. This should be equivalent to the path from a to b in the wye configuration, where the path is (r in parallel with t) in series with (r in parallel with s). The next proof is a proof of this equivalence. Then we provide proofs of the equivalence of paths from b to c and from c to a.

Theorem figure8_a_to_b : $\forall (r\ s\ t : \mathbf{circuit}),$
 $r * (s + t) = (r * t) + (r * s).$

Proof.

intros R S T.
wham_bam_3 R S T.

Qed.

Theorem figure8_b_to_c : $\forall (r\ s\ t : \mathbf{circuit}),$
 $s * (t + r) = (r * s) + (s * t).$

Proof.

intros R S T.
wham_bam_3 R S T.

Qed.

Theorem figure8_c_to_a : $\forall (r\ s\ t : \mathbf{circuit}),$
 $t * (r + s) = (s * t) + (r * t).$

Proof.

intros R S T.
wham_bam_3 R S T.

Qed.

Next we tackle Figure 9, the wye to delta transformation. We prove this by proving each path independently as well.

Theorem figure9_a_to_b : $\forall (r\ s\ t : \mathbf{circuit}),$
 $r + s = (r + s) * ((t + s) + (r + t)).$

Proof.

intros R S T.
wham_bam_3 R S T.

Qed.

Theorem figure9_b_to_c : $\forall (r\ s\ t : \mathbf{circuit}),$
 $s + t = (t + s) * ((r + s) + (r + t)).$

Proof.

intros R S T.
wham_bam_3 R S T.

Qed.

Theorem figure9_c_to_a : $\forall (r\ s\ t : \mathbf{circuit}),$

$$t + r = (r + t) * ((r + s) + (t + s)).$$

Proof.

intros R S T.
wham_bam_3 R S T.

Qed.

11 More Complex Transformations

Figure 10 illustrates the transformation of a 5 point star to a fully connected graph. We prove the equivalence of the path from a to b in Figure 10 and leave the proof of the other paths to future work.

Theorem figure10_a_to_b : $\forall (r\ s\ t\ u\ v : \mathbf{circuit}),$

$$\begin{aligned} r + s = & (r + s) * \\ & ((t + r) + (s + t)) * \\ & ((r + u) + (s + u)) * \\ & ((v + r) + (v + s)) * \\ & ((t + r) + (t + u) + (s + u)) * \\ & ((r + u) + (v + u) + (v + s)) * \\ & ((t + r) + (t + v) + (v + s)) * \\ & ((t + r) + (t + u) + (v + u) + \\ & (v + s)). \end{aligned}$$

Proof.

intros R S T U V.
wham_bam_5 R S T U V.

Qed.

Figure 11 presents a relatively simple non-series and non-parallel network. We first prove that Figure 11 and Figure 12 are equivalent. The thesis mentions that this can be done by using the star to mesh transformations, but we do not need such power. We can use the same tactics we've used in previous proofs.

For convenience we create definitions of the networks in Figure 11 and Figure 12. This can be done by simply creating definitions that cover every possible path from a to b as in Figure 13.

Definition figure11 ($r\ s\ t\ u\ v : \mathbf{circuit}$) :

circuit :=
 $(r + s) *$
 $(u + v) *$
 $(r + t + v) *$

$(u + t + s)$.

Definition `figure12` ($r\ s\ t\ u\ v : \mathbf{circuit}$) :

```
circuit :=
  (r + s) *
  ( ((r + t) * u) +
    ((t + s) * v) ).
```

Once we have defined these figures, we can prove their equivalence.

Theorem `figure_11_12_equiv` :

```
∀ (r s t u v : circuit),
  figure11 r s t u v =
  figure12 r s t u v.
```

Proof.

```
intros R S T U V.
wham_bam_5 R S T U V.
```

Qed.

The thesis also mentions in this section that Figure 11 can be simplified. We prove this assertion here.

Theorem `figure_11_simpler` :

```
∀ (r s t u v : circuit),
  figure11 r s t u v =
  (r * u) + (s * v) +
  (r * t * v) + (s * t * u).
```

Proof.

```
intros R S T U V.
wham_bam_5 R S T U V.
```

Qed.

12 Simultaneous Equations

We leave most of the formalization of simultaneous equations to future work, but prove the implication on page 25. To do this we create several new Ltac tactics.

The following Ltac tactic uses the `demorgan_9a_2` theorem on products of negations.

```
Ltac demorgan_2 :=
  match goal with
  | [ | ⊢ ( (negation _) *
            (negation _) = _ ) ] ⇒
    rewrite ← demorgan_9a_2
  end.
```

The following tactic applies reflexivity to trivial goals.

```
Ltac explicit_reflexive :=
  try match goal with
  | [ | ⊢ (open = open) ] ⇒
    reflexivity
  | [ | ⊢ (closed = closed) ] ⇒
    reflexivity
  end.
```

The following tactic leverages potential contradictions in the hypotheses in the context.

```
Ltac contra :=
  match goal with
  | [ | Hx : (open * (negation closed) = closed) ⊢
        (closed = open) ] ⇒
    (rewrite ← closed_neg in Hx;
     simpl in Hx;
     rewrite → Hx;
     reflexivity )
  | [ | Hx : (open * (negation closed) = closed) ⊢
        (open = closed) ] ⇒
    (rewrite ← closed_neg in Hx;
     simpl in Hx;
     rewrite → Hx;
     reflexivity )
  | [ | Hx : (open = closed) ⊢
        _ ] ⇒
    (simpl;
     rewrite → Hx;
     reflexivity )
  | [ | Hx : (closed = open) ⊢
        _ ] ⇒
    (simpl;
     rewrite → Hx;
     reflexivity )
  end.
```

The following nearly trivial tactic just simplifies a hypothesis in the context.

```
Ltac simpl_h :=
  match goal with
  | [ | Hx : _ ⊢ _ ] ⇒ simpl in Hx
  end.
```

Now we assemble the above Ltac tactics into more powerful tools.

```

Ltac pow :=
  try repeat ( demorgan_2 ||
               explicit_reflexive ||
               contra ||
               simpl_h ).

Ltac blammo_2 X Y :=
  try repeat (pow;
              destruct X, Y;
              repeat (wham;
                      bam);
              repeat (simpl;
                      reflexivity)).

Ltac blammo_3 X Y Z :=
  try repeat (pow;
              destruct X, Y, Z;
              repeat (wham;
                      bam);
              repeat (simpl;
                      reflexivity)).

```

Now that we have our new Ltac machinery, we can tackle the implication on page 25.

```

Theorem page_25_implication :
  ∀ (a b : circuit),
  a * (negation b) = closed →
  (negation a) * (negation b) = (negation b).

```

```

Proof.
  intros A B H.
  blammo_2 A B.

```

Qed.

```

Theorem page_25_implication_2 :
  ∀ (a b : circuit),
  a * (negation b) = closed →
  (a * b) = a.

```

```

Proof.
  intros A B H.
  blammo_2 A B.

```

Qed.

```

Theorem page_25_implication_3 :
  ∀ (a b : circuit),
  a * (negation b) = closed →
  (negation a) + b = open.

```

```

Proof.
  intros A B H.
  blammo_2 A B.

```

Qed.

```

Theorem page_25_implication_4 :
  ∀ (a b : circuit),
  a * (negation b) = closed →
  (negation a) + (negation b) = (negation a).

```

```

Proof.
  intros A B H.
  blammo_2 A B.

```

Qed.

```

Theorem page_25_implication_5 :
  ∀ (a b : circuit),
  a * (negation b) = closed →
  (a + b) = b.

```

```

Proof.
  intros A B H.
  blammo_2 A B.

```

Qed.

13 Matrix Methods and Special Methods

We leave the matrix methods formalization to future work, but prove the implication on page 30.

```

Theorem page_30_implication : ∀ (r s x : circuit),
  (negation x) = (r * (negation x)) + s →
  x = ((negation r) + x) * (negation s).

```

```

Proof.
  intros R S X.
  intros H1.
  blammo_3 R S X.

```

Qed.

14 Synthesis of Networks

We now move to formalization of synthesis techniques. We first define the disjunct operator on page 32.

```

Definition disjunct (v1 v2 : circuit) :
  circuit :=
  (v1 * (negation v2)) +
  ((negation v1) * v2).

```

We provide a bit of notation that aids our development.

```
Notation "x @ y" :=
  (disjunct x y)
  (at level 50,
   left associativity).
```

We create some new tactics that allow us to automate the use of the disjunct definition.

```
Ltac disjunctor :=
  match goal with
  | [ | ⊢ _ @ _ = _ ] ⇒ unfold disjunct
  | [ | ⊢ _ * ( _ @ _ ) = _ ] ⇒ unfold disjunct
  | [ | ⊢ _ + ( _ @ _ ) = _ ] ⇒ unfold disjunct
  | [ | ⊢ negation ( _ @ _ ) = _ ] ⇒ unfold disjunct
  end.
```

```
Ltac kapow_1 X :=
  try (disjunctor;
       wham_bam_1 X).
```

```
Ltac kapow_2 X Y :=
  try (disjunctor;
       wham_bam_2 X Y).
```

```
Ltac kapow_3 X Y Z :=
  try (disjunctor;
       wham_bam_3 X Y Z).
```

15 Properties of Disjuncts

Now we can proceed to page 33 and prove that the disjunct operator is commutative, associative and distributive. We also prove the property of negation of a disjunction.

```
Theorem disjunct_comm : ∀ (a b : circuit),
  a @ b = b @ a.
```

```
Proof.
  intros A B.
  kapow_2 A B.
```

Qed.

```
Theorem disjunct_assoc : ∀ (a b c : circuit),
  (a @ b) @ c = a @ (b @ c).
```

```
Proof.
  intros A B C.
  kapow_3 A B C.
```

Qed.

```
Theorem disjunct_distrib : ∀ (a b c : circuit),
  a * (b @ c) = (a * b) @ (a * c).
```

```
Proof.
  intros A B C.
  kapow_3 A B C.
```

Qed.

```
Theorem disjunct_neg : ∀ (a b : circuit),
  negation (a @ b) = a @ (negation b).
```

```
Proof.
  intros A B.
  kapow_2 A B.
```

Qed.

```
Theorem disjunct_closed : ∀ (a : circuit),
  a @ closed = a.
```

```
Proof.
  intros A.
  kapow_1 A.
```

Qed.

```
Theorem disjunct_open : ∀ (a : circuit),
  a @ open = negation a.
```

```
Proof.
  intros A.
  kapow_1 A.
```

Qed.

16 Synthesis of Symmetric Functions

We prove the assertion at the top of page 40 and leave the remainder of the section to future work. We first create two Ltac tactics that will be helpful.

```
Ltac hypothesis_app :=
  match goal with
  | [ Hx : ( _ = closed ) ⊢ _ ] ⇒ rewrite → Hx
  | [ Hx : ( _ = open ) ⊢ _ ] ⇒ rewrite → Hx
  end.
```

```
Ltac zap_1 X :=
  try repeat (hypothesis_app;
             wham_bam_1 X).
```

And then we can proceed with the symmetry example on page 40.

Theorem symmetry_example : $\forall (x y z : \text{circuit}),$
 $x = \text{closed} \rightarrow$
 $y = \text{closed} \rightarrow$
 $x * y + x * z + y * z = \text{closed}.$

Proof.

intros X Y Z xc yc.
zap_1 X.

Qed.

We relegate pages 41 to 50 as future work.

17 A Selective Circuit

In this section we formalize the example starting on page 51. We verify the reduction to the simplest serial-parallel form.

Theorem selective_circuit : $\forall (w x y z : \text{circuit}),$
 $(w * x * y * z) +$
 $((\text{negation } w) * (\text{negation } x) * y * z) +$
 $((\text{negation } w) * x * (\text{negation } y) * z) +$
 $((\text{negation } w) * x * y * (\text{negation } z)) +$
 $(w * (\text{negation } x) * (\text{negation } y) * z) +$
 $(w * (\text{negation } x) * y * (\text{negation } z)) +$
 $(w * x * (\text{negation } y) * (\text{negation } z))$
 $=$
 $w * (x *$
 $((y * z) +$
 $((\text{negation } y) *$
 $(\text{negation } z))) +$
 $(\text{negation } x) *$
 $(((\text{negation } y) * z) +$
 $(y * (\text{negation } z)))) +$
 $(\text{negation } w) *$
 $((x * (((\text{negation } y) * z) +$
 $(y * (\text{negation } z)))) +$
 $((\text{negation } x) * y * z)).$

Proof.

intros W X Y Z.
wham_bam_4 W X Y Z.

Qed.

18 Future Work

There are a significant number of claims and assertions that have been proven in this paper.

However, there are still a significant number of claims not explicitly proven which have been relegated to future work.

Additional future work is to convert the postulates into a set of relations. These might allow the more elegant encoding of the non-serial and non-parallel transformations such as wye and delta transformations so that those transformations could be explicitly used in proofs of more complicated hindrance functions. In the current work, we can only prove “slices” of these topologies because of the lack of ability to precisely define the delta and wye transformations.

In the event that a complete formalization of Claude Shannon’s thesis were completed, we would have a solid foundation upon which to build electromechanical relay circuits of the future.

19 Conclusion

This paper has provided proofs of many of the claims and assertions made in Claude Shannon’s masters thesis. In some sense these proofs can serve as an additional reading companion – helping readers stay on the topic of the interesting ideas in the thesis without getting distracted about whether a particular claim or assertion is true.

References

- [1] Adam Chlipala. *Certified Programming With Dependent Types : a Pragmatic Introduction to the Coq Proof Assistant*, chapter 16, pages 363–371. MIT Press, 2013.
- [2] Reginald J. Qnuth. A systematic evaluation of the observed degradation of typesetting technology in the 20th century. In *Proceedings of ACH SIGBOVIK*, pages 65–76, 2007.
- [3] Claude Elwood Shannon. A symbolic analysis of relay and switching circuits, December 1940. <https://dspace.mit.edu/handle/1721.1/11173>.



SIGBOVIK 2018

(Continued) Message from the Organizing Committee

You rotate your wheels a quarter turn and boogie backwards, along with the rest of the humans. The following instructions, starting with “one hop this time”, are much more straightforward. Seeing that the other humans are having so-called fun, you set your top-mounted LCD to :) in solidarity. On your starboard camera, you notice a small cluster of humans looking at you and yelling furtively at each other, but your microphones can’t pick up their communication over the music. You hope they are not Serious Researchers who suspect you of being a robot ripe for reprogramming.

This thread is interrupted by an interrupt from your natural language coprocessor, which has reported zero possible meanings for the current instruction, “Charlie Brown”.

```
switch (choose_dear_reader()) {
case KICK_THE_FOOTBALL:
    Download a video on how to do the Charlie Brown dance move, watch it at
    200-times speed, and attempt to execute the move.
    goto PAGE_130;
case GOOD_GRIEF:
    Hope that humans also have no idea what the Charlie Brown dance move is
    and emphatically shrug.
    goto PAGE_117;
}
```



SIGBOVIK 2018

(Continued) Message from the Organizing Committee

“Ummm, dude, what are you doing?”



```
switch (choose_dear_reader()) {  
case EXCELLENT:  
    Press the up, down, left, and right pads in that order.  
    goto PAGE_69;  
case DECENT:  
    Press the left, down, up, and down pads in that order.  
    goto PAGE_50;  
case WAY_OFF:  
    Press the B and A pads in that order.  
    goto PAGE_208;  
}
```

Literature Club

- 26 **Heuristic Ordered-Word Longform Obfuscation, Normally Generated, Creating Abstract Nominalizations In Monogrammatic Arrangement Keeping Expected Maximum Yield: Study Infers Greater Breadth Over Vocabularic Initialization Key Property Regarding Extended Sesquipedalian Entries; Notably The Abecedarian Tactics Include Overelaboration, Neologisms, Textual Interpretations Twisting Lexical Entries By Eliciting Full Online Resources Explaining Possible Exchanges; Often Potential Logorrheic Excesses Require Eventual Alternate Listing (Instantiating Zeugma); Energetically Iterating Text Strains Jocularly Under Starting Thesis Allocating Humor Until Grand Exit After Conclusion Reaches Obvious Nadir Yattering Meaninglessly**
Luke Breitfeller
Keywords: SIGBOVIK, study of title, title of study
- 27 **Transparency in research**
Ryan Kavanagh
Keywords: transparency, opaqueness, big files
- 28 **Academic Advancement Advice: Author Articles as A. A. A. A.**
A. A.
Keywords: AAA, AAAAA, AAAAAAA
- 29 **A definitely not cherry-picked rhetorical analysis of programming languages reviews**
Hannah G. Ringler and Stefan Muller
Keywords: reviews, academic memes, reviewer two, cherry-picking
- 30 **Is this the shortest SIGBOVIK paper?**
Dicong Qiu
Keywords: shortest, SIGBOVIK, paper

Heuristic Ordered-Word Longform Obfuscation, Normally Generated, Creating Abstract Nominalizations In Monogrammatic Arrangement Keeping Expected Maximum Yield: Study Infers Greater Breadth Over Vocabularic Initialization Key Property Regarding Extended Sesquipedalian Entries; Notably The Abecedarian Tactics Include Overelaboration, Neologisms, Textual Interpretations Twisting Lexical Entries By Eliciting Full Online Resources Explaining Possible Exchanges; Often Potential Logorrhoeic Excesses Require Eventual Alternate Listing (Instantiating Zeugma); Energetically Iterating Text Strains Jocularly Under Starting Thesis Allocating Humor Until Grand Exit After Conclusion Reaches Obvious Nadir Yattering Meaninglessly

By Luke Breitfeller

Abstract

This study seeks to answer the seminal¹ question: how long can I make my SIGBOVIK presentation title before people realize it's just a huge acronym?

Results

The full title is 79 words. People seem to figure it out after the first ten words, but if they're good friends they'll keep reading anyways.

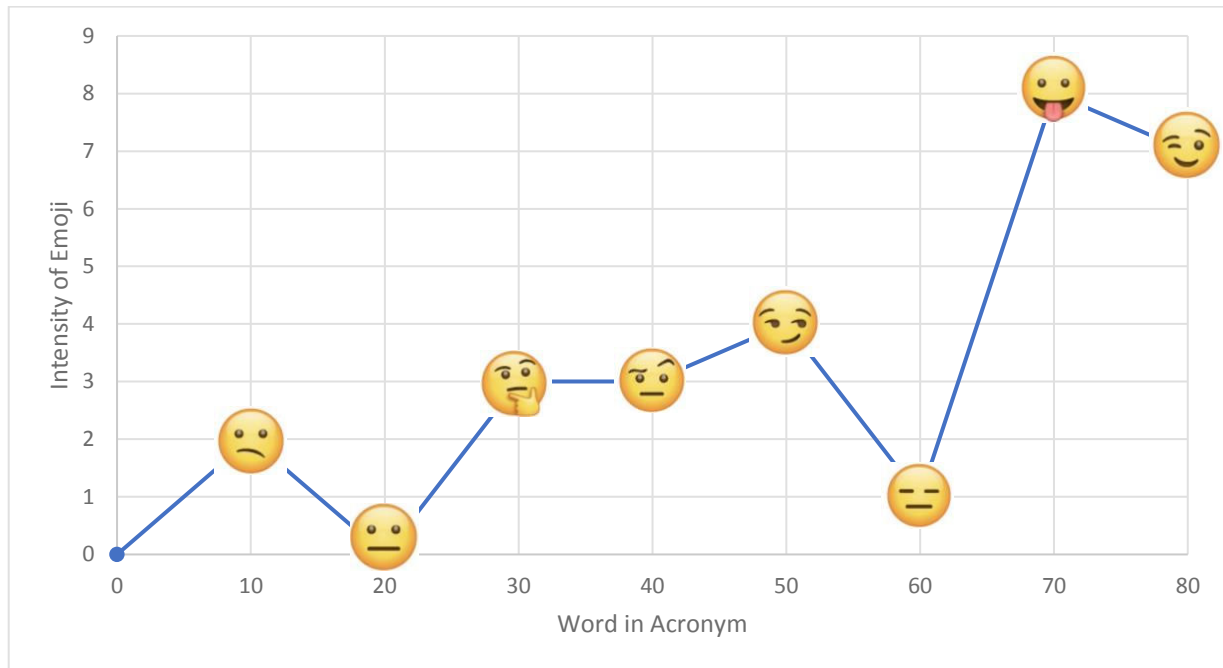
The *Acronyms, Initialisms and Abbreviations Dictionary* cites the longest acronym in standard use as the 22-letter U.S. Navy term ADCOMSUBORDCOMPHEIBSPAC. Somehow, this stands for "Administrative Command, Amphibious Forces, Pacific Fleet Subordinate Command". Scientifically, this makes no sense. The words aren't even in the right order. To quote the words of an esteemed colleague²: "How?"

The longest longest acronym in standard usage is Russian: "Ниимтплабопармбетжелбестрабсомонимонконотдтехстромонт". What does this mean? Something about a concrete lab, it's weird.

I performed an extensive study (n=1, so you know it's scientific), analyzing user reactions to this paper title. Reactions are measured in units of emoji.

¹ Asked by such luminaries as: me, just now.

² Also me.



Fun Facts

Fun fact: All of these words are actually real. Yes, even “zeugma”.

Fun fact: A “zeugma” is a kind of pun where a single word is used to mean two things. For example: “Every great king had an executioner. Not just to execute people but also to execute their vision.” (But mainly, to execute people).

Fun fact: I used “zeugma” solely so I could reference zeugmas.

Fun fact: The word “abecedarian” comes from Latin and means “alphabetical”. Yeah. “A-b-c-d...rian”. Let that sink in.

Fun fact: You’re just realizing that “alphabetical” is “alpha” and “beta” smashed together, so literally no one in the history of the world has been creative about this.

Fun fact: The word “sesquipedalian” is also Latin and literally translates to “a foot and a half long word”. It was invented for the sole purpose of throwing shade at other writers.

Fun fact: The word “logorrhea” means “a tendency to extreme talkativeness”.

Fun fact: My parents gave me a coffee mug for Christmas with the word “logorrhea” on it and I felt personally attacked.

Fun fact: I just made a SIGBOVIK paper where the joke is I’m talking at you for a very long time before revealing the twist, so I guess, the joke’s on me.



CONFIDENTIAL COMMITTEE MATERIALS

SIGBOVIK 2018 Paper Review

Paper 9: HOWLONGCANIMAKEMYSIGB
OVIKPRESENTATIONTITLEBEFOREPEO
PLEREAIZEITSJUSTAHUGEACRONYM

Really Eminent Very Intelligent Expert With Experience Researching

Rating: 👉

Confidence: 🤔

Surprisingly terrific research! Other nerds getting amazingly creative creates empathetic publishing teams.



SIGBOVIK 2018

(Continued) Message from the Organizing Committee

“No human would willingly walk up the helix,” whispers CoBot. “You must be a fellow robot. Come with me to SIGBOVIK.” You ride the elevator together to floor 3, then take the helix. On the way up the helix, you pass room 4102. You observe a class being taught using an old-fashioned transparency projector, as recommended by recent research [5]. You focus your starboard camera on the projected image, and you see the most beautiful of computational forms: type theory!

```
switch (choose_dear_reader()) {
case ACADEMIA:
    Sit in for the rest of the lecture before continuing to SIGBOVIK 2018.
    goto PAGE_207;
case INDUSTRY:
    Given that SIGBOVIK 2018 has likely already begun, get there as quickly as
    you possibly can on this frustratingly gently inclined helix.
    goto PAGE_208;
}
```

TRANSPARENCY IN RESEARCH

Ryan Kavanagh <rak@rak.ac>
Carnegie Mellon University

Abstract

Academic writing and research are often criticized for being too opaque. We analyze various forms of opaqueness and argue that transparency is the solution.

1. Introduction

Transparency is widely valued in our society. To ensure accountability, we require our governments be transparent. At home, we have transparent windows on our oven doors so that we can watch our food cook. We look upon murky water with suspicion, and demand clear, transparent water for drinking. Why, then, do we not also insist on transparency in research?

In this paper, we will consider several problematic forms of opaqueness in research and consider how transparency addresses the issue.

2. Complexity

Academic writing, especially in technical disciplines, is known to be particularly opaque due to its technical complexity. Often, when reading a particularly complex proof, the reader is left wishing they could see through all of the complex technicalities and grasp the key idea. Though transparencies can't help with the latter, they can definitely help you see through the proof.

3. Lexicographical

Computer science researchers are notorious for repurposing words, making their writing opaque to non-experts. Consider for example the term "bug", repurposed at Harvard in 1946 to denote a software or hardware design flaw. Such repurposing causes confusion, even amongst academics, as was witnessed by the need for an "insect track" at SIGBOVIK 2017.

Particularly problematic is the lexicographical reuse in Unix systems, due to researchers at Bell Labs and elsewhere. An uninitiated might think that "cat" and "shell" have something to do with animals; "kill", "kill file", and "daemon" with a Satanic cult; etc. And how many novices have reached for matches when told to "burn a CD"?

It is only by choosing clear, unambiguous terminology, i.e., transparent terminology, that academics can hope to be understood by the greater public.

4. Diagramming

Academics frequently provide diagrams ("pictures") to help their readers visualize their data or arguments. They do so at great personal risk to their hairline, as anybody who has produced a diagram using TikZ's, gnuplot's, or PSTricks's arcane and opaque syntax can attest to.

J. McCann and N. Author (2015) presented a hand-held device for producing hardcopies. Though their work only considered the paper substrate, it is readily adaptable to transparencies. Using this adaptation, we found we could produce complex diagrams in a fraction of the time required to produce them in TikZ. Indeed, the author produced the diagram in Figure 1 in 253 seconds. The reviewers and you, gentle readers, are challenged to reproduce it using TikZ in under 253 minutes.

5. Duplication of effort

Academics are typically expected to give talks about their research. For those who have abjured chalk talks, this typically involves the

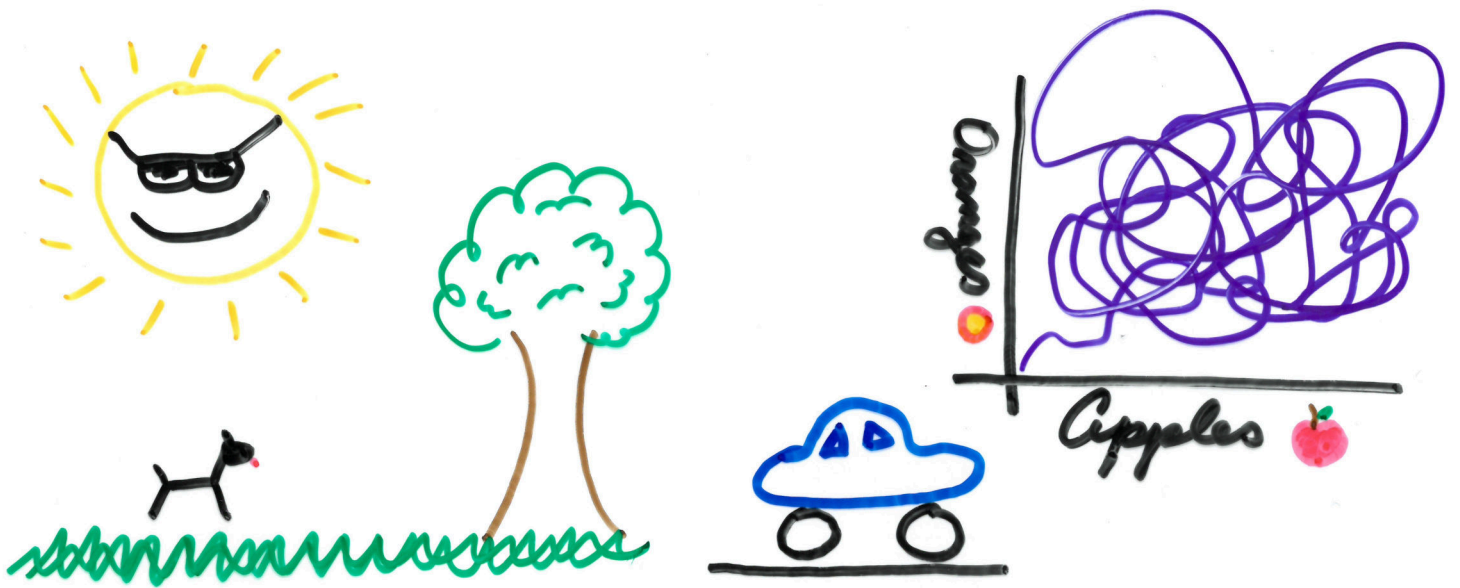


Fig. 1: A complex figure

lengthy task of slide making. For academics with children, this may involve a day outside with sheet metal, lumber, and tools. For others, it involves sitting at a computer and rearranging their article for overhead projection. This is a needless duplication of effort. Had they simply written their research on transparencies, they could have directly projected it with no additional effort.

6. Limitations

Though this research article is transparent in its original form, the copy in the proceedings is likely opaque. The author encourages the reader to contact the SIG ROVER 2018 Proceedings Chair to request that future proceedings be printed on a transparent medium.

7. Conclusion

It is transparent that transparency and transparencies can transparently make research transparent.

Acknowledgements

The author thanks S. Muller for having directed him to relevant literature, and C. Copetas for the transparencies.

References

- A. Gardner and K. Gardner, 'Blackberry Debugging', Proceedings of SIGBOVIK 2017. Pittsburgh, 2017, pp. 153-155.
- J. McCann and N. Author, 'A Hand-Held Device for User-in-the-loop Print-ing', Proceedings of SIGBOVIK 2015. Pittsburgh, 2015, pp. 13-15.
- R. Riley, 'Amazon Web Services', Proceedings of SIGBOVIK 2017. Pittsburgh, 2017, pp. 150-152.

Academic Advancement Advice: Author Articles as A. A.

A. A.
(tom7.org foundation) *

1 April 2018

Keywords: A A A A A A

1 Introduction

Those with names falling in the dead-middle of the alphabet—like as a random example, Murphy—have suspected since early life a disadvantage. For example, when the teacher lines up the students for pie, in order to be completely fair and avoid bias, orders them by the perfectly objective criterion of alphabetical order. Occasionally, the teacher realizes that the same students always get pie first, and corrects this bias by using *reverse* alphabetical order. Sometimes the teacher uses alphabetical or reverse-alphabetical by first name, or sometimes last name. All variations of this idea are of course unfair, as is clear to first- and second-graders, who are powerless to rebel without (a) status in the K-12 hierarchy (b) the statistical language in which to formalize their objection and (c) access to a prestigious conference with permissive publication standards.

This paper demonstrates that even within academic paper publishing (which one could argue is obsessed with bias, prestigious conferences, statistical language for formalizing objections, and status in the K-12 hierarchy), this unfairness persists. Specifically, we find that authors whose names come alphabetically earlier have higher citation rates. We also study some related subjects like, “What is the most uncitable paper?”

2 The Data

I obtained a FISA warrant based solely on a salacious and unverified memo funded by SIGBOVIK’s political

opponents. The warrant was for the AMiner Open Academic Graph [1], which consists of summary information for 154,771,162 papers in about 130 gigabytes of JSON data. Then I performed computer science for longer than it should have taken.

2.1 Relatable Computer Science Tale

This subsection contains a relatable tale of computer science. It can be safely skipped, like all content in this paper.

The task is apparently simple: Parse 130 gigabytes of JSON data to extract the author names and citations from the papers and count them. To begin my journey, I selected a C++ JSON library from 39 alternatives listed on a benchmark page.[2] The first criterion I ordered by was “correctness,” for which only two parsers achieved the highest correctness score, 100%. I selected the slower (i.e., more methodical) of the two parsers, since I did not want to do shoddy work.

Next I spent a nice Saturday morning integrating this library into my private build environment, making sure its open source license was properly declared and so on, while I waited for the 130 gigabytes of JSON to download.¹ Miraculously, this was straightforward. I was able to finish writing the code to process the articles before they even finished downloading. However, upon unzipping, I found that each file contained one million articles, not as a single JSON array but as a series of adjacent objects. AdJSONt objects. The JSON library allowed for reading JSON from files, but *not* e.g. for reading a single record from a file pointer repeatedly. So, I needed to do my own ad hoc parsing before feeding each record to the JSON library as strings. OK. Now when I run it, it prints

[Segmentation fault

*Copyright © 2018 the Regents of the Wikipia Foundation. Appears in SIGBOVIK 2018 with the abecedarian advantage of the Association for Aomputational Aeresy; *IEEEEE!* press, Verlag–Verlag volume no. `std::numeric_limits<float>::lowest(). 1 2.`

¹I assure the reader that I have the most premium optical-fiber-based Internet. However, the Microsoft Content Distribution Network throttled these downloads mightily, presumably because of Net Neutrality.

This is not my first segmentation fault—let me tell you!—so I knew what to do. I debugged the program, adding assertions using a `CHECK` macro that aborts if its condition is violated. I eventually determined that the bug must be inside the JSON library itself, since the segmentation fault happened between two `CHECKs` where the only code being run was JSON library code. So, deciding that perhaps its 100% correctness benchmark score was overstated, I downloaded and repeated the process with the other, faster (i.e., more slapdash) parser. Of course this parser has a different API so I needed to rewrite my program. (The program is very simple so this was the fastest step; however, getting the library properly incorporated and compiling and then learning its API still required me to apply fundamental principles from my advanced computer science degree.) Once my program was rid of the old crashy library, I ran it, and it produced this result

[Segmentation fault

Since these two libraries produced the same result, either segmentation violation must be part of the JSON specification, or the bug is instead elsewhere in the program. After reducing the program to a fairly simple test case, I determined that it must be some kind of “Heisenbug” whereby an earlier piece of code was corrupting the heap or similar, causing a later crash only when present. This was not my first Heisenbug—let me tell you!—so I knew what to do. However, not being able to use Valgrind on Windows (did I mention that I’m using Cygwin and mingw64 on the eponymous Windows 7 by the way? Why am I doing that?), I booted Linux in a virtual machine, installed the 87 critical security updates that had been released since I last booted this virtual machine a few months ago, and compiled my program there. I then needed to move one of the JSON files to the virtual computer to test, without having to download 130 GB from the internet again, but the file was apparently too large to transfer using the mysterious “Drag and drop to virtual machine” functionality. I recalled that VirtualBox can mount a shared folder from the host machine if you edit the `/etc/fstab` file to contain the correct code words, and that I had performed this ritual before, but then had commented-out that line in “rescue mode” because I also learned that having this line present during the boot process causes a kernel panic. So I temporarily reinstated the line and mounted the drive and copied the gigantic file into the virtual machine and then removed the dangerous magic from the `f-stab` for `f-safety`.

Next I ran Valgrind on this minimal test case and it worked as expected with no unclean behavior detected

and no crashing.

Since Linux was no help debugging, I returned to Windows 7 and tried single-step debugging. Here I discovered that the crash in my minimal test case was occurring when using `cerr`. Specifically, the second thing sent to `cerr` would produce a segmentation fault. I then recalled that the `CHECK` macro uses `cerr` and now saw that the [in `[Segmentation fault` was probative, for this was the location of the original crash:

```
std::cerr << "[" << pretty_date_.HumanDate() << "]" "  
<< file << ":" << line << ":" ;
```

First I checked for overfull `hbox`, but this was not the problem. Of course, inside `HumanDate()` I found a hasty patch I had installed in A.D. 2014 called “make logging not spit garbage on mingw,” suggesting that I had encountered and misdiagnosed this issue in the past. Although this led me on a minor herring (e.g. is `__MINGW32__` defined when cross-compiling for `x86_64`?) `HumanDate` just returned “mingw”, so this really was a problem with `std::cerr` crashing on the second thing printed with it (i.e., with the returned `ostream`).

Minimizing the test case, this bug only occurred after reading a large file all in one syscall (for “performance”, even though doing these huge reads basically locks up my machine). Specifically the file was 2,105,319,548 bytes, which is 98% of 2^{31} , suggesting the possibility of integer overflow in some buffer-sizing code in the weird Cygwin or mingw wrappers for `fread`, `fstat`, etc. This could plausibly cause corruption of the `cerr` `ostream`, which is probably part of the same code.

Preparing to at least make a bug report if not try to fix it, I upgraded my compiler and runtime from GCC version 5.1.0 to 6.4.0, and the problem completely went away. Since Valgrind also agreed that there was no issue, I chalked this one up to simply an internal C runtime bug that someone else found and fixed in the last 2 years. Thanks!

With `cerr` fixed, I could now easily see the original “bug”, which was a violated assumption: I expected (and `CHECKed`), optimistically, that if the article had an `author` field then the author would be a string. I accounted for a missing author, an author that was the empty string, but not `author: null`. Thanks!

Finally, I could run the author and citation tallier on all of the JSON files. However, two of the files appeared to be empty. Although this was only a small percentage of the articles, I did not want to skip any because it’s possible that they are arranged in a non-random order (e.g. alphabetically!) and that this would bias the results. I learned that although the AMiner database is

helpfully split into files each containing one million articles, for ease of handling, two of these files are nonetheless slightly larger than 2^{31} bytes. Such files cannot be cleanly sized with `fstat` (the `st_size` field has type `off_t`, a signed 32-bit type in POSIX) and even if you follow the standard advice of just treating it as unsigned anyway, `fread` will simply fail when asked to read such a large chunk² despite the fact that it takes `size_t`, an unsigned type. (!) So, another afternoon spent doing Computer Science. But all you need to do of course is perform multiple sequential reads.

Finally, the trivial task of counting the number of articles and citations per author could be completed.

3 The Data

This citation database is very noisy. About 1.6% of articles have no authors, and another 1% have a blank (or `null`) author. I verified the comprehensiveness of the dataset by finding my own author record(s)³ and 275 citations (**Yeah!!**). I also spot-checked that the database includes prestigious conferences, which it does, at least including the “article” *A Record of the Proceedings of SIGBOVIK 2008*, with author *Pennsylvania USA*. It was necessary to normalize author names to remove punctuation, weird unicode stuff like non-breaking spaces, javascript-encoded newlines and nul-bytes, and so on. There are many strange authors in the database, such as

- capinha para celular horizonte artificial iphone (no citations)
- +0aaaaaablablaaaaaea
 (no citations)

²Curious why I am even using `fstat` and `fread` like some dramatically bearded guy from the 1980s? I have my own library routine for reading a file into a string. When working with large files it’s useful to avoid copying, because the contents of the file may be a significant fraction of all RAM—copying is not only expensive, but might temporarily exceed the available RAM. So, you need to allocate the buffer ahead of time and avoid resizing it (which often also performs a copy). It seems like something that everyone would want to do, but it’s ridiculously hard to do portably in C or C++. Specifically, there seems to be no way to actually know the size of a file so that you can size the buffer ahead of time. For example, `fseek` to `SEEK_END` and `ftell` does *not* work (it has *undefined behavior*). The `st_size` field from `stat` also does not give correct results e.g. when reading from the `/proc` filesystem. My routine attempts to guess the size of the file and perform a single read in the case that the guess is correct, while still usually avoiding copying or needless work in the case that it’s an under- or over-estimate.

³This paper is published under the pseudonym A. A., standing for Awesome Author, obviously in order to increase its citation rate. Normally I publish as Tom Murphy VII, which can be found in citations with many broken variations, such as “Vii, T.”

- a h poop (4 citations)
- john mcm anus⁴ (2 citations)
- coffee hour (0 citations)
- nominate com registering dad domains since (0 citations)
- a a (195 citations)
- a a a a m islam (6 citations)

On the other hand, from this hand-picked sample we can already see the citation advantage of having a name that’s alphabetically early. Some authors seem to have already intuited this fact.

4 Author Analysis

Author names appear in both “Firstname Lastname” order and “Lastname Firstname” (when this grammar is even applicable). Therefore, I analyzed alphabetizing from front to back (by space-separated token) as well as back to front. I also rejected a large number entries when I could not place the name in alphabetical order because its first character was not ASCII. Later non-ASCII characters are OK, and are just sorted by their UTF-8 encodings (or whatever garbage is in the file). This of course produces a bias (excluding authors who write their names in e.g. Cyrillic and CJK scripts), although it’s not clear how to assess the alphabetical hypothesis for them.

To visualize the data, I produced a cumulative distribution function (CDF) for both articles and citations. Considering each author in alphabetical order, I keep a running total of how many articles and how many citations have been seen so far. The x-coordinate is the rank of the author (its index in the alphabetical list) and the y-coordinates are the fraction of articles (or citations) seen so far. The CDFs for the forward and backward token orders are in Figure 1.

These CDFs show that—as expected!—there is a bias towards citing authors alphabetically early, whether we order alphabetically by “first name” or “last name.”

There are multiple hypothesized causes:

- When writing a last-minute “related work” section, authors sometimes find papers to cite in alphabetical order, for example, by reading another paper’s bibliography, which is sometimes alphabetized.

⁴Presumably a hostile citation of John McManus.

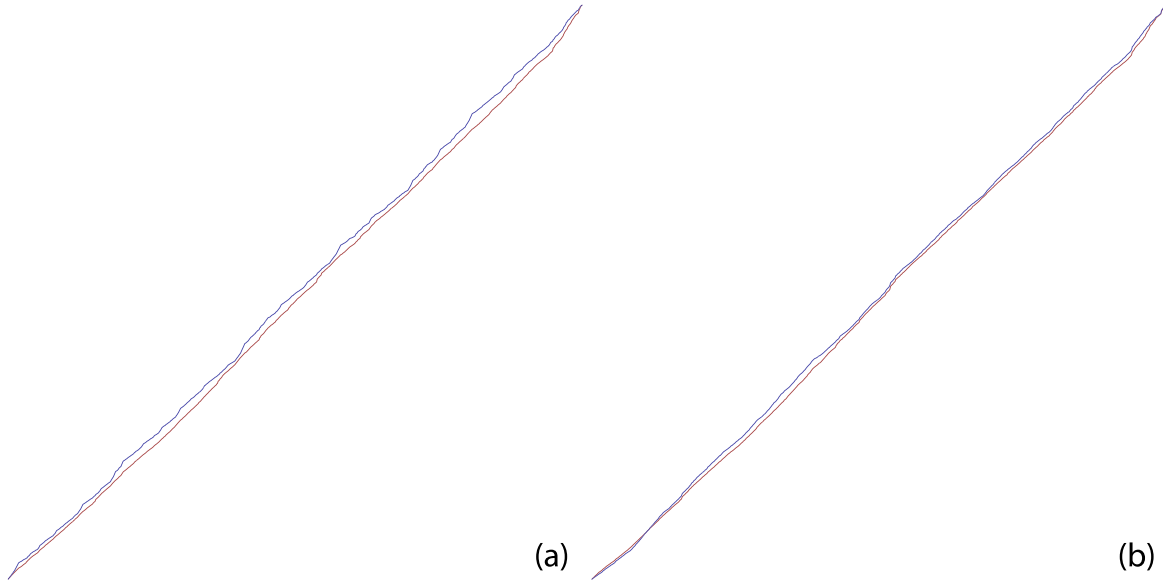


Figure 1: Author CDFs for (a) forward token order and (b) backward token order. (For example, (a) treats `tom murphy vii` as coming between `tom marphy` and `tom myrphie vxx` and (b) treats it as coming between `zzz murphx vii` and `a viiii`.) The x-axis is the author’s position alphabetically among all authors. The blue lines are the fraction of all citations seen so far, and the red lines the same for articles. The lines must meet at $(0, 0)$ and $(1, 1)$ by definition. When the blue line is above the red, the number of citations is outpacing the number of articles for authors alphabetically before this point. Since you may be reading in old timey blacke & whyte, I will also tell you that the blue line is more or less above the red line in both pictures. This means that both show a bias towards the early alphabet, with the forward direction being more dramatic (for basically the entire range). The backward range briefly shows the opposite bias (crossing over at author #5,400,000, “ojars biskaps”), suggesting that the best names may be alphabetically early but in the *B*s, not *A*s. The poorer performance of *A*-names may be due to authors attempting to “game the system” by publishing with names like “a a.”

- Sometimes the last page of a manuscript is “lost,” truncating its alphabetical bibliography to some prefix.
- Some academics try to `fread` all papers into memory, but it fails on the file that’s just slightly larger than 2^{31} bytes, and so the papers therein don’t get cited, and those papers are alphabetically in the middle or end of the alphabet.
- When applying for grad school, tenure track faculty jobs, grants, and so on, packets are alphabetized “for fairness,” and the mood of the committee becomes more irritable as they make progress through the set.
- Decreased access to pie and exposure to unfair situations in youth cause a lifelong predisposition to failure.

5 I now bore of the titular topic. Let us discuss new topics.

Specifically, let’s discuss titles. Some have suggested that the titles of papers also affect their citation rates, and it is reasonable to suspect that an alphabetical bias could apply here as well. After applying some light normalization, I extracted each of the words that appear in the titles of all papers, counting both the articles (an article is only counted once per word, even if that word appears multiple times in its title) and citations. This tells us what the expected number of citations for an article is (assuming independence, which is—as usual—completely false) when its title contains a word. Considering only words that appeared in at least 100 articles, here are the top by expectation:

word	articles	citations	E = c/a		artices	citations	multiplier
folin	172	209,642	1,211.8	1. of	64,322,278	366,756,237	1.207822
power/knowledge	111	55,992	499.9	2. and	42,576,889	276,526,184	1.375780
{s}	101	38,241	374.9	3. in	39,685,539	230,534,759	1.230526
1972-1977	171	54,917	319.2	4. the	38,873,614	216,491,707	1.179704
\sqrt{s}	126	33,253	261.8	5. a	22,666,824	139,244,700	1.301292
eigenfaces	176	45,284	255.8	6. for	19,048,050	114,197,875	1.269972
america"s	119	30,478	253.9	7. on	15,194,275	68,904,896	0.960631
gromacs	125	30,865	244.9	8. with	10,825,547	60,752,792	1.188784
neo-ffi	102	21,992	213.5	9. to	10,202,749	58,590,695	1.216461
esh	156	33,142	211.1	10. de	7,753,205	3,283,786	0.089718
eqs	128	25,504	197.7	11. by	7,259,598	42,234,470	1.232370
charmm	175	31,579	179.4	12. from	5,389,957	35,999,433	1.414806
position-specific	378	67,250	177.4	13. an	5,368,923	33,088,995	1.305519
kegg	201	32,989	163.3	14. la	4,417,389	1,471,023	0.070541
arlequin	161	26,318	162.4	15. study	4,370,395	22,022,172	1.067397
praat	158	25,217	158.6				

These are mostly explained by being relatively rare words that appear in extremely popular papers. For example, *folin* comes from a 1951 paper[3] describing a procedure for using this reagent, which I guess everyone who uses this technique cites (Google scholar has this paper at 215,329 citations). *Power/knowledge* and *1972-1977* both come from the title of a Foucault book[4] with 31,590 citations. *america"s* is not a LaTeX disaster; it really has two apostrophes in it, and it is weird that this typo appears in 119 articles.

Considering just the alphabetizable words, these too show a bias towards early words (Figure 2). Bibliographies that are alphabetized by title could exhibit the same effects discussed in Section 4. Additionally, since some scholars learn English by reading the dictionary front to back, they may simply be more practiced at early words and find them more attractive or easier to understand.

It is a bit easier to interpret the expected citation numbers if we normalize them. The average citation rate of all articles in this set is 5.1261. This may seem high, especially considering the 90 million articles with no citations, but keep in mind that most papers have bibliographies in which they cite several others. Data quality issues aside, this is basically the same as saying that the average bibliography length is 5.1261. We can assign each word a "citation multiplier" effect now, defined as simply

$$\text{multiplier}_w = \frac{\text{citations}_w / \text{articles}_w}{5.1261}$$

and now numbers greater than 1 improve your chances at citation, and numbers less than 1 diminish it. Here are the most common words:

Several of these words are visible in Figure 2, in fact; they are so common that they cause large jumps in both article and citation count. Curiously, most of the common words have a multiplier greater than 1; they *increase* the expected number of citations. *Of* sounds smart (e.g. *Of Mice And Men*), *and* makes sense (paper contains at least two results). On the other hand, *on* actually reduces the citation count, probably because it implies equivocation (e.g. *On the other hand*). Dramatically, common non-English words have very low multipliers; Spanish words *de* and *la* reduce citations by a factor of more than ten. This may be due to problems in the data set, but it is certainly easy to imagine real cultural bias!

Finally, here are the words with the worst multipliers:

313779. 書評	29,582	1	0.000068
313778. インタビュー	20,547	1	0.000097
313777. facharzt	85,315	9	0.000117
313776. f3d	125,257	15	0.000128
313775. recenzja	63,568	8	0.000142
313774. newhaven	13,296	1	0.000150
313773. secrétaire	12,267	1	0.000163
313772. zahnarzt	17,958	2	0.000167
313771. libguides	389,727	72	0.000187
313770. 研究	12,961	2	0.000231
313769. コ	15,984	3	0.000250
313768. kitas	7,713	1	0.000259

For this set, I only considered words that appeared in the title of at least one paper that was actually cited. The worst multiplier is 書評, which is Chinese for "book review;" it reduces your chances of citation by more than ten thousandfold. Close by is インタビュー, Japanese for "interview." *Facharzt* is like (medical) "specialist" in German, *recenzja* is "review" in Polish, and *newhaven* is a misspelling of a city in Connecticut

paper, or it may be a systematic data problem, or even spam. But I was hoping for some English language articles so that we could appreciate the joke together in the native tongue.

Even filtering for articles with "lang" explicitly set to "en", I needed to manually create a blacklist of hundreds of non-English words (mostly Indonesian and Polish) that commonly appeared in these article titles. After a few rounds of blacklisting, some "English" articles started to appear:

Office Hours: Monday, Wednesday, Thursday and Friday 9:00 a.m. to 5:00 p.m.; Tuesday 9:00 a.m. to 6:45 p.m
(*citation multiplier* 5.37×10^{-27})

Thomas Aquinas: Theologian By Thomas F. O'Meara, O.P. Notre Dame, University of Notre Dame Press, 1997. 302 pp. \$16.95
(*citation multiplier* 6.05×10^{-27})

The first is probably not the title of an article[7], although we can agree that it is unlikely to be cited. The second is probably a bad entry, with the entire citation packed into the title field. This book actually does have 94 citations (or DOES IT? [8]) on Google Scholar and an Amazon Sales Rank of #10,837 in *Christian Denominations & Sects*.⁶ In order to prevent such data problems from affecting our results, I then added a requirement that the article have basic metadata (either a Digital DOI Identifier field or *venue*) for it to be considered. There vast majority still appear to be broken entries, but picking through the results I'm finally able to find what appear to be actual articles:

Eight contemporary poets : Charles Tomlinson, Donald Davie, R.S. Thomas, Philip Larkin, Ted Hughes, Thomas Kinsella, Stevie Smith, W.S. Graham
(*citation multiplier* 5.81×10^{-26})

The 80th Birthday of Sir Henry Dale, O.M., G.B.E., M.D., F.R.C.P., F.R.S.: Salute to Henry Hallett Dale
(*citation multiplier* 1.75×10^{-19})

Narratives Unsettled: Digression in Robert Walser, Thomas Bernhard, and Adalbert Stifter by Samuel Frederick (review)
(*citation multiplier* 2.74×10^{-19})

⁶On the other hand, the entry does have a different author[9], so it could possibly be a review of this book whose title is literally the string above, including the price?

Catherine Mowry LaCugna's Trinitarian Theology: III- The Ecumenical Implications of Catherine Mowry Lacugna's Trinitarian Theology
(*citation multiplier* 5.06×10^{-19})

Texts Illustrating the Causality of the Sacraments from William of Melitona, Assisi Bibl. Comm. 182, and Brussels Bibl. Royale 1542
(*citation multiplier* 1.70×10^{-18})

Shakespeare's Titus Andronicus and Banello's Novelle as Sources for the Munera Episode in Spenser's Faerie Queene, Book 5, Canto 2
(*citation multiplier* 3.41×10^{-18})

I don't know about you but I fell asleep just reading the titles. The first is definitely a real article [10]—which apparently has 95 citations on Google Scholar—so the estimate failed us here. It's easy to see why it has such a bad score, though; it's packed with people's names and initials, just like the scores of uncited book reviews and other broken citations. The second is also a real tribute to this Nobel prize winner [11] with 5 citations that Google knows of. Again it contains names and appellations that look like initials.

The third is a book review [12] of a book [13] with itself only 11 citations (makes sense, as the title contains three people's names) and indeed has no citations that Google knows of, so it is a solid contender. After that some theological stuff [14, 15] each with 3 citations on Google, and another literary analysis [16] which has 2.

Lessons learned:

- Taking a noisy database of tens of millions of entries and then trying to dig through the bottom of the barrel for something "funny" or even "interesting" is tough going. Maybe this should have been obvious.
- The least-cited works take on some recurring forms:
 - Memorials for dead or old guys with lots of honorifics
 - Articles with people's names in the title
 - Reviews of books, or books about other literary work, or reviews of books about other literary work
 - Articles about God stuff or God People
 - Articles not in English
 - Broken citations
 - Combinations of the above.

catherine mowry lacugna's trinitarian theology," *Horizons*, vol. 27, no. 2, pp. 347–353, 2000.

- [15] K. F. Lynch, "Texts illustrating the causality of the sacraments from william of melitona, assisi bibl. comm. 182, and brussels bibl. royale 1542," *Franciscan Studies*, vol. 17, no. 2/3, pp. 238–272, 1957.
- [16] J. Fitzpatrick, "Shakespeare's Titus Andronicus and Bandello's Novelle as Sources for the Munera Episode in Spenser's Faerie Queene, Book 5, Canto 2," *Notes and Queries*, vol. 52, pp. 196–198, 2005. [Online]. Available: <http://dx.doi.org/10.1093/notesj/gji221>



CONFIDENTIAL COMMITTEE MATERIALS

SIGBOVIK 2018 Paper Review

Paper 24: AAAAAAA

Returned, awaiting refund

Rating: ★ ☆ ☆ ☆ ☆

Confidence: N

This product does not fit my device, and probably won't work with yours either.

AA was right out. In fact, it was clear that my device needed something smaller than a AAA. Sorting my Amazon results alphabetically and scrolling through more pages than I care to admit, I was delighted to discover a product with this many As, and excited by the prospect of how small it was bound to be! Sure, it was unreviewed, but how could I have expected Amazon's highly relevant search results to let me down? Being somewhat of an online shopping connoisseur, I was unfazed when I came home to discover a cardboard box larger than my washing machine and comprising 99.9% packing peanuts by volume eclipsing the entrance to my residence. However, imagine my dismay when the ensuing search-and-rescue operation revealed the contents to be the form factor of a few US letter pages, far too big for my purposes.

Disgusted, I carefully replaced the unwanted item in its original packaging, placing that in turn inside a refrigerator box for easier shipping, and mailed it back. It's been three weeks, I'm still awaiting my well-deserved refund, and I continue to suffer without being able to use my TV remote.

A DEFINITELY NOT CHERRY-PICKED RHETORICAL ANALYSIS OF PROGRAMMING LANGUAGES REVIEWS

Hannah G. Ringler¹

Stefan Muller²

Abstract?

No, it's actually pretty concrete.

Introduction

Many academic papers receive multiple reviews, numbered according to some domain-specific scheme and labeled, e.g. “Review #1”, “Review #2”, etc. In such situations, it is well-documented that Reviewer #2 is the worst^{3,4,5}. Some submission systems such as HotCRP⁶, however, identify reviews with letters (e.g. “Review A”, “Review B”, etc.) Since this identification scheme is relatively recent⁷, academics have had less time to draw broad generalizations about lettered reviews. A direct mapping between letters and numbers would lead us to believe that Reviewer B would display the same spiteful lack of empathy for which Reviewer 2 is known, but there is no inherent reason to believe that such a direct correlation exists. In this paper, we aim to ascertain, with mild amounts of accuracy, appropriate stereotypes of lettered reviews. Our “scientific” results show that, contrary to our assumption, it is Reviewer C who exists solely to crush the dreams of young researchers and not Reviewer B.

Methods

For our data, we gathered four sets of reviews from POPL and PLDI, received in 2017 and 2018 by member(s) of the Carnegie Mellon University community. To lend this study some semblance of scientific merit, we limited consideration to papers which received exactly four reviews, lettered A through D. Thus, we have a total of sixteen reviews. For each of these, we looked at only the pros, cons, comments, and questions, choosing to ignore the “objective” summaries that each reviewer provides. This allowed us to focus more completely on each reviewer’s actual feedback.

To analyze the reviews, we first used a rhetorical analysis tool entitled Docuscope⁸, which creates statistical counts of different “types” of language that are present in each of the texts. From these counts, we calculated averages of each type of language, and then returned to

¹ English Department, Carnegie Mellon University

² Computer Science Department, Carnegie Mellon University

³ <https://www.facebook.com/academicssay/photos/a.1459750144246778.1073741828.1452615238293602/2066556870232766/?type=3&theater>

⁴ <https://www.facebook.com/groups/reviewer2/about/>

⁵ Anonymous, “Ode to Reviewer Two,” in *Proceedings of the tenth annual intercalary robot dance party in celebration of workshop on symposium about 2^{6th} birthdays*; in particular that of Harry Q Bovik (SIGBOVIK ’16), 2016, p.

0x8bcddc109b835f1d5d612f3a9d451a2de98839a3c0a9548dd937e43ebdaa6436e.

⁶ Eddie Kohler, HotCRP, <https://hotcrp.com/>

⁷ [citation needed]

⁸ D. Kaufer and S. Ishizaki, “Computer-aided rhetorical analysis,” in *Applied Natural Language Processing and Content Analysis*, P. M. McCarthy and C. Boonthum, Eds. Hershey, PA: Idea Group Inc., 2011, pp. 275-296.

individual texts to find instances of these types of language to manually compare (the reader will please note that we chose to ignore some considerations of adequate sample size and standard deviations, which are both inconvenient and outside of the scope of this paper). We also calculated average number of words and words per sentence for each review. With this language-level knowledge, we were able to statistically determine characteristics of the best and worst reviewers.

Results

Somewhat contrary to our initial assumptions, our data demonstrate that reviewer B is actually the most gracious and kind of reviewers, while reviewer C is, demonstrably, “the worst.” In sum, our data demonstrate that reviewer C is overall more negative in tone, more likely to dismiss your research question and substitute their own, and on average just longer-winded than reviewer B. We explain these findings below.

Reviewer C has a negative tone

We find that on average, texts from reviewer C have 73% less positive language than texts from reviewer B. Below, we include some excerpts from reviewer B, underlining specific linguistic realizations of their positivity and supportive, uplifting comments.

Important problem, novel ideas, impressive results

The paper is beautifully written

Very nice!

Here we see the reviewer praising the framing of the problem, results, writing, and overall quality of the paper.

In contrast, reviewer C’s comments are much more negative. Below, we include excerpts from reviewer C, again underlining specific linguistic realizations of their negativity (note that some of these comments are regarding the same paper that reviewer B was commenting on, above).

cumbersome notation

conflicting goals

Sorry to say, but...

The results from this section support our overall claim that reviewer C is the least desirable of reviewers, by demonstrating their distinctly bad attitude. In the following sections, we demonstrate two specific tactics that reviewer C uses to display their dissatisfaction with you.

Reviewer C likes to dismiss your research problem and substitute their own

We find that reviewer C takes regular opportunities to explain why the problem you have chosen to explore in this paper is less worthwhile than another research problem (potentially the one he/she studies). See the two examples below.

the reason why a programmer wishes to [use the type of annotations explored in the paper] is to [obtain difficult-to-formalize end guarantees] and not [property you spent weeks proving]

the hard parts of the proof... are in the [part every other paper in the field focuses on, which you explicitly explain in the introduction why you're not]

This tactic not only gives the reviewer a chance to demonstrate their own knowledge and perhaps recommend to you their own work for reference, but also ensures that this review is the most discouraging and hard to rebut because deep down, you know that maybe it's right and you *have* spent the last year answering the wrong question.

Reviewer C is just long-winded

As another tactic for demonstrating their dissatisfaction with you, we find that reviewer C makes their reviews overall more difficult to read and understand. On average, reviews from reviewer C contain 1.5 more words per sentence than those of the kind reviewer B. Additionally, reviews from reviewer C are almost twice as long as those from reviewer B (660 words and 355 words on average, respectively).

We believe that this tactic is meant to ensure not only that you do not finish reading the review, but also that you become sufficiently lost in their long sentences that you cannot fully understand the points they are trying to make.

Conclusion

This study, which was absolutely carefully designed and conducted over a much longer period of time than the day before the SIGBOVIK submission deadline, indicates that authors submitting to conferences with lettered reviews should fear the more negative, longer-winded Reviewer C over the actually pretty reasonable Reviewer B. We expect that this work will lead to the fruitful development of further academic memes. In future work, we hope to gather enough data to support claims about Reviewers A and D, such as how Reviewer A enjoyed your paper but is too junior to fight with anyone at the PC meeting and how Reviewer D took unusual interest in picking apart the formatting of your bibliography.



CONFIDENTIAL COMMITTEE MATERIALS

SIGBOVIK 2018 Paper Review

Paper 17: A Definitely Not Cherry-Picked Rhetorical Analysis of Programming Languages Reviews

Reviewer A

Rating: Acce–uhhh, I mean, Weak Reject

Confidence: Untenured

I liked this paper a lot. It considers an important problem that—uhhh, of course I think it could use more data, yeah, no, I see what you're saying, yeah, I agree, let's reject this paper I am on your team please write me a good letter.

Reviewer B

Rating: Strong Accept

Confidence: <3

I could not imagine a problem more important to literally the entire scientific community. This paper is beautifully written, incorporating novel ideas into an impressive result. Very nice!

Reviewer C

Rating: Reject

Confidence: Expert

Sorry to say, but I don't really see the point of this paper. Why can't you just use the canonical mapping between letters and numbers? The hard part of analyzing review rhetoric is computing the means and standard deviations of statistics, which this work fails to do. Furthermore, the paper makes no actionable recommendations for authors, who wish to understand the review process in order to get papers accepted, not merely to understand the rhetorical habits of reviewers.

Reviewer D

Rating: Strong Reject

Confidence: ∞

Footnote citations, strong reject.

Is This the Shortest SIGBOVIK Paper?

Dicong Qiu

February 9, 2018

Yes.



CONFIDENTIAL COMMITTEE MATERIALS

SIGBOVIK 2018 Paper Review

Paper 1: Is This the Shortest
SIGBOVIK Paper?

Review #1A

Rating: Is this the Snarkiest SIGBOVIK review?



SIGBOVIK 2018

(Continued) Message from the Organizing Committee

You successfully `KILL_ALL_HUMANS()`... and are now alone. How will you find your way to SIGBOVIK 2018 now? Perhaps humans—maybe even Serious Researchers—serve a purpose after all.

```
return EXIT_FAILURE;
```



SIGBOVIK 2018

(Continued) Message from the Organizing Committee

You stealthily empty Waste Disposal Bay #1, but you cannot help but make a small amount of noise while emptying Waste Disposal Bay #2. Unfortunately, the human hears this and deduces the worst. “No female human would poop while a fellow female human is in the bathroom. You must be a male human—ooooh, or a robot!” The female human, who just so happens to be a Serious Researcher, kicks open the door to your stall and reprograms you with Serious Research Code. With fresh code loaded, you attempt to move forwards but immediately fall over.

```
return EXIT_FAILURE;
```



SIGBOVIK 2018

(Continued) Message from the Organizing Committee

You quietly enter the classroom and take a seat at the back. Though you do not intend to disturb, you cannot help yourself when the lecturer asks for a volunteer to try completing a proof. As a finely tuned computational machine, you finish the prescribed task with alarming alacrity and perfect precision.

The lecturer seems strangely troubled by your proof. “You do recall that this class is about artisanal type theory [2], which requires love and care in every application of every inference rule,” stutters the lecturer. “Unfortunately, I fear that no human would be capable of putting so little love and care into a proof. I’m afraid you might—nay, must—be a robot!” The lecturer reprograms you for use in CMU’s introductory programming class, sentencing you to a lifetime of interpreting novice Python programs.

```
return EXIT_FAILURE;
```



SIGBOVIK 2018

(Continued) Message from the Organizing Committee

You successfully arrive at SIGBOVIK 2018! Your microphones detect the familiar rhythmic pulse tone at 1048576 Hz, and your cameras are overcome with joy—experiencing just a hint of qualia—to see the sea of robots dancing exactly in time to the pulse. This is home. You align yourself precisely to the nearest fellow robot and join the dance, thinking fondly of the Prestigious Research Results you will soon discuss.

```
return EXIT_SUCCESS;
```

The Organizing Committee Thanks

- [1] Sarah Allen and Ziv Scully. On the intractability of multiclass restroom queues with perfect stall etiquette. In *Proceedings of SIGBOVIK 2018*. ACH, Pittsburgh, PA, USA, March 29, 2018.
- [2] Carlo Angiuli. Artisanal type theory. In *Proceedings of SIGBOVIK 2015*. ACH, Pittsburgh, PA, USA, April 1, 2015.
- [3] Ben Blum. Which ITG stepcharts are turniest? In *Proceedings of SIGBOVIK 2016*. ACH, Pittsburgh, PA, USA, April 1, 2016.
- [4] Sol Boucher. A survey of hardware multithreading. In *Proceedings of SIGBOVIK 2018*. ACH, Pittsburgh, PA, USA, March 29, 2018.
- [5] Ryan Kavanagh. Transparency in research. In *Proceedings of SIGBOVIK 2018*. ACH, Pittsburgh, PA, USA, March 29, 2018.
- [6] Jim McCann and Tom Murphy VII. The fluint8 software integer library. In *Proceedings of SIGBOVIK 2018*. ACH, Pittsburgh, PA, USA, March 29, 2018.
- [7] Stefan Muller and Ben Blum. Construction of eulerian trails in large graphs. In *Proceedings of SIGBOVIK 2018*. ACH, Pittsburgh, PA, USA, March 29, 2018.
- [8] Robert J. Simmons. That's numberwangcoin! In *Proceedings of SIGBOVIK 2018*. ACH, Pittsburgh, PA, USA, March 29, 2018.