the association for computational heresy

presents

a record of the proceedings of

SIGBOVIK 2020

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

cover art by chris yu

global chaos courtesy of sars-cov-2

carnegie mellon university

pittsburgh, pa

april 1, 2020
SIGBOVIK

A Record of the Proceedings of SIGBOVIK 2020

ISSN 2155-0166

April 1, 2020

Copyright is maintained by the individual authors, though obviously this all gets posted to the Internet and stuff, because it’s 2020.

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 2020
Message from the Organizing Committee

For generality’s sake, we have templatized the message of the organizing committee. The actual message may be produced by running the TeX command at the end.

\newcommand{\Message2020}{
% TODO: generalize ordinal indicators

Friends, family, colleagues, acquaintances for whom we have not yet overcome the activation energy to engage with regularly, and strangers doomed to the same fate: The Association of Computational Heresy welcomes you to the #1\textsuperscript{th} annual meeting of the Special Interest Group on Harry Q#2 Bovik in celebration of Bovik’s #3\textsuperscript{th} birthday. In lieu of data about the submission and review process this year, we encourage you to ponder the perils of empiricism as well as our innovations in the publishing process: most notably, triple-blind peer review. In plain English, we have the following definition.

\textbf{triple-blind peer review} /ˈtrɪpl-blaɪnd piːr rɪˈvjuː/ noun

1. Scholarly peer review that minimizes bias by concealing not only the identities of the authors and reviewers from each other, but also by concealing the papers from the reviewers. Compare: single- and double-blind peer review.

However, let us undertake some performative formalization. Given a set of names of authors (auth) and reviewers (rev), we consider the role of author and reviewer as, under the Curry-Howard correspondence, proofs of the following higher-order linear logic propositions/processes conforming to the following protocols.

\[\vdash P :: (c : \exists_{a:auth,p:paperBy(a)} \forall_{r:rev} reviewOfBy(p, r) \to \text{bool})\]
\[\vdash Q :: (d : \forall_{a:auth,p:paperBy(a)}\exists_{r:rev} reviewOfBy(p, r))\]

Under duality, it is clear that communication between $P$ and $Q$ is sound. Then, each level of blindness (single to triple) is achieved by successively abstracting the definition of rev for $P$ then auth and paperBy for $Q$ respectively; we refer to the work of Harper and Lillibridge [1] on translucent
sums to achieve this. Higher-order notions of blind review require a stronger metatheory like a linear temporal-linear logical framework (that’s TWO linears!); we encourage future work to investigate this idea. Now that the general chair has redeemed himself for not submitting a paper this year, we would like to thank our authors and reviewers for their phenomenal work and for adjusting to the new review process as well as the continuous effort of volunteers who have made this year’s conference possible, which include but are not limited to: Chris Yu for the cover art, Catherine Copetas for managing our finances and other administrative concerns, and Ryan Kavanagh for organising the organisers. Moreover, the program chair also thanks Rose Bohrer and Stefan Muller for further advice. Lastly, we would like to thank Sol Boucher for assembling the proceedings as well as Jenny Lin and Siva Somayyajula for never working in various capacities (see figure 1).

The SIGBOVIK 2020 Organising Committee
Pittsburgh, PA

Jenny Lin (easy chair)
Siva Somayyajula (generalized hard-ass chair)
Sol Boucher (fold-out couch)
Rose Bohrer (reclining chair)
Ryan Kavanagh (rockin’ chair)
Chris Yu (swivel chair)
Stefan Muller (ergonomic office chair)

\Message2020{14}{uarantine} {$2^6$}

References

## Blindsight is 2020

### Reading Skills

1. Anonymous paper
2. State-of-the-Art Reviewing: A radical proposal to improve scientific publication

### SIGBOVIK

3. A thorough investigation of the degree to which the COVID-19 pandemic has enabled subpar-quality papers to make it into the proceedings of SIGBOVIK, by reducing the supply of authors willing to invest the necessary effort to produce high-quality papers
4. Is this the longest Chess game?
5. Optimizing the SIGBOVIK 2018 speedrun
6. Retraction of a boring follow-up paper to “Which ITG stepcharts are turniest?” titled, “Which ITG stepcharts are crossoveriest and/or footswitchiest?”

### Programming Languages

7. Ntuation copassing style
8. Formulating the syntactically simplest programming language
9. Type-directed decompilation of shell scripts
10. Verified proof of P=NP in the Silence theorem prover language

### Systems

11. SIGBOVIK ’75 technical note: Conditional move for shell script acceleration
12. NaN-gate synthesis and hardware deceleration
13. hbc: A celebration of undefined behavior

### Theory

14. A polynomial-time SAT algorithm
15. Empire Machines and beyond
16. Artificial General Relativity
17. Lower gauge theory: Dead duck or phoenix?
18. Making explicit a constant appearing in an impossible equation of Carl and Moroz

### Security

19. Determining the laziest way to force other people to generate random numbers for us using IoT vulnerabilities
20. Putting the ceremony in “authentication ceremony”

### Artificial Intelligence & Machine Learning

21. Image2image neural network for addition and subtraction evaluation of a pair of not very large numbers
22. Robot ethics: Dangers of reinforcement learning
23. Learning to be wrong via gradient ascent: A broken clock is right twice a day, but a clock that runs backwards can be used to tell the time!
GradSchoolNet: Robust end-to-end *-shot unsupervised deepAF neural attention model for convexly optimal (artificially intelligent) success in computer vision research

:Natural Intelligence & Human Learning

: Sorting with human intelligence
: Synthesizing programs by asking people for help online
: HonkFast, PreHonk, HonkBack, PreHonkBack, HERS, AdHonk and AHC: The missing keys for autonomous driving

: Education

: A disproof of the Theorem of Triviality
: Visualizing player engagement with virtual spaces using GIS
: How to git bisect when your test suite is having a bad day

: Blockchain Gang

: RegisToken

:The SIGBOVIK 2020 Field Guide to Conspiracy Theories

: Faking ancient computer science: A special SIGBOVIK tutorial
: MORSE OF COURSE: Paper reveals time dimension wasted
: The Pacman effect and the Disc-Sphere Duality

: Serious Business

: The SIGBOVIK 2020 field guide to plants
: Deep industrial espionage
: What is the best game console?: A free-market–based approach

: Funny Business

: Can a paper be written entirely in the title? 1. Introduction: The title pretty much says it all. 2. Evaluation: It gets the job done. However, the title is a little long. 3. Conclusion: Yes.
: Erdős-Bacon-Sabbath numbers: *Reductio ad absurdum*
Reading Skills

1 Anonymous paper
Anonymous Author

Keywords: anonymous, triple blind, review

2 State-of-the-Art Reviewing: A radical proposal to improve scientific publication
Samuel Albanie, Jaime Thewmore, Robert McCraith, João F. Henriques

Keywords: peer review, SotA, State-of-the-Art, computer vision, machine learning
Dear Author,

Thank you very much for your submission 'Anonymous Paper'.

We are sorry to inform you that your paper was rejected. The venue received a great many submissions of high quality. Each submission was thoroughly and extensively reviewed by our expert panels. After week-long online and offline discussions, we selected a small subset of papers to be accepted. Unfortunately, your paper was not one of them.

We have enclosed detailed reviews of your paper below. We hope that these will help you in your scientific work and look forward to your future submissions!

Reviewer 2

Generally speaking, this is interesting work which may well find some readers in the community. However, in its present form, the paper suffers from a number of issues, which are outlined below.

Abstract

This paper, titled 'Anonymous Paper,' one more time addresses the problem of Anonymity in SIGBOVIK. The approach is novel, and is well-described in the paper. Yet, it is only incremental with respect to the state of the art. This especially holds when compared against the latest works of Anonymous Author. The approach is shown to work only in theory; but despite its obvious limitations, it is shown to have potential under specific circumstances.

Area

SIGBOVIK is not new. The past decade has seen a flurry of papers in the area, which would indicate progress and interest. However, the impact outside of SIGBOVIK has been very limited, as is easily reflected in the ever-decreasing number of citations. Unfortunately, SIGBOVIK finds itself in a sandwich position between a more theoretical community and a more practical community, which can both claim a bigger intellectual challenge and a greater relevance in practical settings. This is especially true for the field of Anonymity, where the last big progress is now decades ago. As refreshing it feels to finally again see progress claimed in the field, the present paper must be read and evaluated in this context.

Originality
The central problem of the paper is its novelty with respect to related work. Just last month, this reviewer attended a talk on Anonymous Paper by Anonymous Author at Anonymous Institution where a very similar solution to the exact same problem was presented. The paper is actually online at

http://scigen.csail.mit.edu/cgi-bin/scigen.cgi?author=Anonymous%20Author

Please relate your work carefully against this existing work, precisely highlighting the advances.

**Approach**

The approach is novel in the sense that the exact same approach has not been examined before.

A central problem with the approach is that the difference with respect to earlier work is simply too small. An incremental approach such as this one may be appreciated in practice, as it may easily integrate into existing processes; but for a research paper, the community expects bigger leaps.

The paper suffers from an excess of formalism. Several important definitions are introduced formally only, without a chance to clearly understand their motivation or usage. The authors should be encouraged to make more use of informal definitions, building on the well-established formal definitions in the community.

It is good to see the approach scale to real-life settings. The price of scalability, however, is having to cope with a multitude of details, which the paper only glosses over, without ever providing a complete picture. The authors would be better served to provide a tangible abstraction of their approach; maybe it only works in limited settings, but at least, for these, we can understand how and why.

The authors should be happy to see their approach being used outside of academia. However, this challenges the novelty of the present submission: Not only is the approach is no longer superior to the state of the art, it even is not longer superior to the state of the practice - ironically, because it now defines what the current state of practice is. The submission had better be framed as an experience report and target non-academic readers.

**Evaluation**

As nice as it is to see the approach defined and its properties discussed, it has to be evaluated whether it works on actual subjects. Without a detailed evaluation, we can never know whether the claimed properties also hold under less abstract and less favorable circumstances.

**Limitations**

The section on 'threats to validity' pretty much sums everything up: The approach cannot claim external validity (no surprise, given the evaluation results); on top, the authors themselves list
important threats to internal and construct validity.

The fact that the authors themselves apparently cannot list future work fits into this very picture. This is a clear indication of work being stuck in an impasse; if one ever needed a living proof how Anonymity has become the laughingstock of the SIGBOVIK community, or how SIGBOVIK is more and more becoming a dead branch of science, here it is. With so many threats and limitations, it is unclear whether the paper can be published at all, even in a thoroughly revised form.

Reproducibility

Your code and data are not available as open source. Being unavailable for the general public, there is no way for readers (or this reviewer) to validate your claims, which is a strict requirement for a scientific contribution. Please submit it as an electronic appendix with the next revision.

Presentation

The typography of your paper is a disgrace. Respect the style instructions as given by the publisher. Respect paper lengths. Do not cheat with super-small fonts. Learn what good typography is. Learn how to use LaTeX, and how to use LaTeX properly. Do not use multi-letter identifiers in LaTeX math mode. Distinguish - (hyphen) between elements of compound words, -- (en-dash) in ranges, $$ (minus) for math, --- (em-dash) for digressions in a sentence. use the correct quotes (`` and ''). In BibTeX, capitalize titles properly. Use \dots rather than ... . This reviewer stops here.

The paper has numerous presentation issues. The paper contains numerous typos - Page 2, for instance, has a period '.' instead of a comma ','. This is a sign of careless proofreading, and ultimately disrespect - if the authors do not care about their paper, why should the reader care? At least try to produce a polished version for submission.

Keep in mind that a high number of presentation issues eventually will hinder the readers and reviewers to understand what your work is about. Should you find misunderstandings in the above review, ask yourself what you could have done to avoid these.

Summary

Points in favor: (+) An interesting approach to Anonymity in SIGBOVIK (+) Paper does a good attempt at describing the approach

Points against: (-) Far away from scientific mainstream (-) Incremental (-) Insufficient evaluation (-) Substantial presentation issues

Recommendation

Reject.
Footnote

Generated by autoreject.org
Reviewer: Definitely not the SIGBOVIK webmaster
Rating: 3rd grade reading comprehension
Confidence: Confident that some cool people said to email reviews to the sigbovik-reviews mailing list and not to easychair

<table>
<thead>
<tr>
<th>Conference website</th>
<th><a href="http://sigbovik.org/2020">http://sigbovik.org/2020</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Submission link</td>
<td><a href="https://easychair.org/conferences?conf=sigbovik2020">https://easychair.org/conferences?conf=sigbovik2020</a></td>
</tr>
<tr>
<td>Abstract registration deadline</td>
<td>March 13, 2020</td>
</tr>
<tr>
<td>Submission deadline</td>
<td>March 13, 2020</td>
</tr>
</tbody>
</table>

Note that this is the appropriate way to submit papers but not reviews, triple-blind or otherwise. See the website for details on that process: sigbovik.org/2020

By submitting to SIGBOVIK 2020, I solemnly affirm all of the following:

- My submission does not have page numbers.
- I have the rights to use all images that appear in my submission.
- My submission really does not have page numbers. Not even a little bit.
- Page numbers are only permitted if they are really necessary for some incredibly niche joke. We're dead serious.

Figure 1: The easychair paper submission site

conference date. We welcome prerecorded videos from those who cannot physically make it to SIGBOVIK!

New: To submit your triple blind review, download The Official Incredibly Secure Extra Blind SIGBOVIK Review Template (in LaTeX). After filling out review-template.tex, send just your edited review-template.tex to sigbovik-reviews@lists.andrew.cmu.edu (the included .cls file is just so you can see the compiled result). If you are displeased by

Figure 2: The SIGBOVIK 2020 website instructions for submitting reviews
STATE-OF-THE-ART REVIEWING: A RADICAL PROPOSAL TO IMPROVE SCIENTIFIC PUBLICATION

Samuel Albanie, Jaime Thewmore, Robert McCraith, Joao F. Henriques
SOAR Laboratory,
Shelfanger, UK

ABSTRACT

Peer review forms the backbone of modern scientific manuscript evaluation. But after two hundred and eighty-nine years of egalitarian service to the scientific community, does this protocol remain fit for purpose in 2020? In this work, we answer this question in the negative (strong reject, high confidence) and propose instead State-Of-the-Art Review (SOAR), a novel reviewing pipeline that serves as a “plug-and-play” replacement for peer review. At the heart of our approach is an interpretation of the review process as a multi-objective, massively distributed and extremely-high-latency optimisation, which we scalarise and solve efficiently for PAC and CMT-optimal solutions.

We make the following contributions: (1) We propose a highly scalable, fully automatic methodology for review, drawing inspiration from best-practices from premier computer vision and machine learning conferences; (2) We explore several instantiations of our approach and demonstrate that SOAR can be used to both review prints and pre-review pre-prints; (3) We wander listlessly in vain search of catharsis from our latest rounds of savage CVPR rejections.

1

1 If a decision tree in a forest makes marginal improvements, and no one is around to publish it, is it really “state-of-the-art”?

George Berkeley,
A Treatise Concerning the Principles of Human Knowledge (1710)

1 W.A/W.A/B → Reject. A single heavily caffeinated tear, glistening in the flickering light of a faulty office desk lamp, rolls down a weary cheek and falls onto the page. The footnote is smudged. The author soldiers on.

1 INTRODUCTION

The process of peer review—in which a scientific work is subjected to the scrutiny of experts in the relevant field—has long been lauded an effective mechanism for quality control. Surgically inserted into the medical field by the cutting-edge work of (Ali al Rohawi, CE 854–931), it ensured that treatment plans prescribed by a physician were open to criticism by their peers. Upon discovery of a lengthy medical bill and a dawning realization that theriac was not the “wonder drug” they had been promised, unhappy patients could use these “peer reviews” as evidence in the ensuing friendly legal proceedings.

Despite this auspicious start, it took many years for the peer review protocol to achieve the popular form that would be recognised by the layperson on the Cowley Road omnibus today. Credit for this transformation may be at least partially attributed to the Royal Society of Edinburgh who were among the first to realise the benefits of out-sourcing complex quality assessments to unpaid experts (Spier, 2002).

Effacing the names of these heroic contributors, in a process euphemistically called anonymous review, was a natural progression. Attempts to go further and have the reviewers retroactively pay
for the privilege of reading a now-copyrighted manuscript (at the discounted price of £50) somehow did not catch on, despite the publishers’ best intentions. Peer review (not to be confused with the French tradition of Pierre review, or indeed the spectacle of a pier revue) has since gone from strength-to-strength, and is now the primary quality filtration system for works of merit in both the scientific and TikTok communities.

Still, something is rotten in the state of reviewing. To determine what exactly is causing the smell, our starting point in this work is a critical review of peer review. We begin by highlighting three key shortcomings of the existing system.

**Ability to Scale.** As anyone who has prepared for a tech internship interview knows, scale is important. And so are corner cases. And so is good communication. But the greatest of these is scale. To avoid carelessly ruling out future careers at Google, we therefore demonstrate an appreciation of the critical importance of this phenomenon. Indeed, it is here that we must mount our first attack on peer review: the algorithm is provably \(O(p)\), where \(p\) is the number of peers. To concretise the implications of this runtime complexity, consider the nation of the United Kingdom which occupies a small number of green and pleasant islands 'twixt the United States and Europe. There are, at the time of writing, 814 hereditary peers in the UK who can be called upon as professional peers. Of these, 31 are dukes (7 of which are royal dukes), 34 are marquesses, 193 are earls, 112 are viscounts, and 444 are barons. Many of these, however, do not sit in the House of Lords (an Airbnb property in which peers can be recruited to review documents), and so cannot be relied upon here. Fortunately, the vast majority of the 789 members of the House are instead peerages “#4lyf”—these are ephemeral honours which are somewhat easier to create because they do not require building new humans from scratch from a limited set of eligible bloodlines. Nevertheless, short of a fairly sizeable second “Loans for Lordships” political scandal, it is hard to foresee the kind rapid growth in the peerage ranks that is needed to meet reviewing demand. We also note here a second concern: for various technical reasons\(^2\), only one hereditary position of the house is held by a woman (Margaret, 31st Countess of Mar), which raises questions about not only the *scale*, but also the *diversity* we can expect among the potential reviewing pool.

**Speed.** The mean age of the House of Lords was 70 in 2017. With a lack of young whippersnappers amidst their ranks, how can we expect these venerable statesmen and stateswomen to do the all-nighters required to review ten conference papers when they are only reminded of the deadline with two days notice because of a bug in their self-implemented calendar app? One solution is to ensure that they take care when sanitising date/time inputs across time-zones. But still, the question remains: how long does peer review really take? Since public data on this question is scarce, we turn to anecdotal evidence from our latest round of reviewing. The results were striking. We found that *any conference review paper batch is likely to contain at least one paper that takes at least ten hours to review*. The blame for these “time bombs” lies with both authors and reviewers, since they arise from the combination of: (1) a review bidding process that allows the reviewer access to only the paper title and abstract; (2) authors who write paper titles and abstracts that bear little resemblance to their work. As a consequence of this mismatch, the unsuspecting reviewer may, on occasion, volunteer for a 47 page appendix of freshly minted mathematical notation, ruining their weekend and forcing them to miss a movie they really wanted to see. Of course, the fact that they actively bid on the paper and were therefore *responsible for its assignment* ensures that they feel too guilty to abandon the review. The proof of why this is problematic is left as an exercise for the reader.

**Consistency.** The grand 2014 NeurIPS review experiment (Lawrence & Cortes, 2015) provides some insight into the consistency of the peer review process. When a paper was assigned to two independent review committees, about 57% of the papers accepted by the first committee were rejected by the second one and vice versa (Price, 2014). While these numbers provide a great deal of hope for anyone submitting rushed work to future editions of the conference, it is perhaps nevertheless worth observing that it brings some downsides. For one thing, it places great emphasis on the role of registering at the right time to get a lucky paper ID. This, in turn, leads to a great deal of effort on the part of the researcher, who must then determine whether a given ID (for example 5738\(^3\)) is indeed, a lucky number, or whether they are best served by re-registering. A similar phe-

---

\(^2\)See Sec. A.2 in the appendix for historical conditions under which a peerage would pass to a female heir.

\(^3\)Thankfully, numerology is on hand to supply an answer. “5738: You are a step away from the brink that separates big money from lawlessness. Take care, because by taking this step, you will forever cut off your ways to retreat. Unless it is too late.” (numeroscop.net, 2020)
nomenon is observed in large-scale deep learning experiments, which generally consist of evaluating several random initialisations, a job that is made harder by confounders such as hyper-parameters or architectural choices.

By examining the points above, we derive the key following principle for review process design. Human involvement—particularly that of elderly hereditary peers—should be minimised in the modern scientific review process. In this work, we focus on a particular instantiation of this principle, State-Of-the-Art Reviewing (SOAR), and its mechanisms for addressing these weaknesses.

The remainder of the work is structured as follows. In Sec. 2, we review related work; in Sec. 3, we describe SOAR, our bullet-proof idea for automatic reviewing; in Sec. 4 we develop a practical implementation of the SOAR framework, suitable for popular consumption. Finally, in Sec. 5, we conclude with our findings and justification for why we anticipate swift community adoption.

2 RELATED WORK

2.1 INTEREST IN THE STATE-OF-THE-ART

Since the discovery of art (Blombos Cave Engravings, ca. 70000 BC) there has been a rising interest in this form of expression, and consequently, the state thereof. From the Medici family of Florence to theatre buff King James I, much effort has been dedicated to patronage of the arts, and much prestige associated with acquiring the latest advances. Pope Julius II was keen to raise the papal state of the art to new heights, namely the ceiling, enlisting the help of renaissance main man Michelangelo. The score of Sistine remains competitive in chapel-based benchmarks, and Michelangelo became a testudine martial artist (with the help of his three equally-talented brothers) (Eastman & Laird, 1984).

From early on, the importance of adding depth was appreciated (Studies on perspective, Brunelleschi, 1415), which continues to this day (He et al., 2016). Recently, the critically acclaimed work of Crowley & Zisserman (2014) illustrated how the state-of-the-art can be used to assess the state of art, calling into question the relevance of both hyphens and definite articles in modern computer vision research. Of least relevance to our work, Fig. 1 depicts state-of-the-art developments in the art world.

Figure 2: (Left) The number of PhDs granted annually exhibits exponential growth (figure reproduced from Gastfriend (2015)), (Right) Google retrieved ngram counts of “State of the Art” over the past 200 years of literature. Note that even when the axes are rotated slightly, it remains difficult to preserve an upwards trend. This evidence suggests that either PhDs are becoming exponentially less productive than their predecessors or that the existing reviewing system does not provide sufficient incentivise to use the term “state-of-the-art” in modern manuscripts. Our proposal directly addresses the latter.

2.2 LITERATURE REVIEW

The Grapes of Wrath. In this classic portrayal of the American Dust Bowl, Steinbeck captures the extremes of human despair and oppression against a backdrop of rural American life in all its grittiness. A masterpiece.

Flyer for (redacted) startup, left on a table at NeurIPS 2019 next to a bowl of tortillas. Hastily put together in PowerPoint and printed in draft-mode amid the death throes of an ageing HP printer, this call for “dedicated hackers with an appetite for Moonshots, ramen noodles and the promise of stock options” comes across slightly desperate.

3 METHOD

Science is often distinguished from other domains of human culture by its progressive nature: in contrast to art, religion, philosophy, morality, and politics, there exist clear standards or normative criteria for identifying improvements and advances in science.

In Sec. 1, we identified three key weaknesses in the peer review process: (1) inability to scale; (2) slow runtime and (3) inconsistent results. In the following, we describe the SOAR review scheme which seeks to resolve each of these shortcomings, and does so at minimal cost to the taxpayer or ad-funded research lab, enabling the purchase of more GPUs, nap-pods and airpods.

3.1 STATE-OF-THE-ART REVIEWING (SOAR)

It is well known that the quality of a scientific work can be judged along three axes: efficacy, significance and novelty. Our key insight is that each of these factors can be measured automatically.

Assessing efficacy. Efficacy is best assessed by determining if the proposed method achieves a new SotA (State-of-the-Art). Thankfully, from an implementation perspective, the authors can be relied upon to state this repeatedly in the text. Thus, rather than parsing results table formats (an error-prone process involving bold fonts and asterisks), we simply word count the occurrences of “state-of-the-art” (case insensitive) in the text. It stands to reason that a higher SotA count is preferable.
Moreover, such an approach avoids the embarrassment of realising that one cannot remember what kind of statistical significance test should be applied since all SotA is significant.

**Assessing significance.** Significance is measured by efficacy. Thus, the efficacy term is weighted twice in the formula.

**Assessing novelty.** The assessment of novelty requires close familiarity with prior art and an appreciation for the relative significance of ideas. We make the key observation that the individuals best placed to make this judgement are the author themselves since they have likely read at least one of the works cited in the bibliography. We further assume that they will convey this judgement by using the word “novel” throughout the document in direct proportion to the perceived novelty of the work.

With the strategies defined above, we are now in a position to define the SOAR score as follows.

\[
\text{SOAR Score} \triangleq \sqrt[3]{S_{\text{SotA}}} \cdot S_{\text{SotA}} \cdot S_{\text{novelty}} / 10. \tag{1}
\]

Here \( S_{\text{SotA}} \) and \( S_{\text{novelty}} \) represent the total occurrences in the manuscript of the terms “state-of-the-art” and “novel”, respectively. In both cases, we exclude the related work section (it is important to avoid assigning SotA/novelty credit to the paper under review simply because they cite SotA/novel work). A geometric mean is used to trade-off each factor, but note that a paper must be both SotA and novel to achieve a positive SOAR score. Lastly, we attach a suffix string “/10” to every SOAR score. This plays no role in the computation of the score.

Note that several factors are not assessed: vague concepts like “mathematical proofs” and “insights” should be used sparingly in the manuscript and are assigned no weight in the review process. If the proof or insight was useful, the authors should use it to improve their numbers. SotA or it didn’t happen.

A key advantage of the SOAR formula is that it renders explicit the relationship between the key scientific objective (namely, more State-of-the-Art results) and the score. This lies in stark contrast to peer review, which leaves the author unsure what to optimise. Consider the findings of Fig. 2: we observe that although the number of PhDs granted worldwide continues to grow steadily, usage of the term “State-of-the-Art” peaked in the mid 1980’s. Thus, under peer review, many PhD research hours are invested every year performing work that is simply not on the cutting edge of science. This issue is directly addressed by measuring the worthiness of papers by their state-of-the-artness rather than the prettiness of figures, affiliation of authors or explanation of methods.

With an appropriately increased focus on SotA we can also apply a filter to conference submissions to immediately reduce the number of papers to be accepted. With top conferences taking tens of thousands of submissions each typically requiring three or more reviewers to dedicate considerable time to perform each review, the time savings over an academic career could be readily combined to a long sabbatical, a holiday to sunny Crete, or an extra paper submission every couple of weeks.

### 4 Implementation

In this section, we outline several implementations of SOAR and showcase a use case.

#### 4.1 Software Implementation and Complexity Analysis

We implement the SOAR algorithm by breaking the submission into word tokens and passing them through a Python 3.7.2 `collections.Counter` object. We then need a handful of floating-point operations to produce the scalar component of Eqn. 1, together with a string formatting call and a concatenation with the “/10”. The complexity of the overall algorithm is judged reasonable.
In defense of revisiting Adapting Adaptations: Are convolutions convolutional enough?

Novel Neville

Submitted on 31 Jul 2019

In recent years, the humble convolution has drawn praise from friends and foes alike for its enviable equivariance, parameter sharing and strong theoretical connection to Joseph Fourier. But is the convolution "convolutional" enough? This question forms the basis of the current work, in which we highlight scenarios in which one does not simply "convolve" a standard convolutional operator, willy-nilly, with all desired inputs.

4.2 WETWARE IMPLEMENTATION AND COMPLEXITY ANALYSIS

In the absence of available silicon, SOAR scoring can also be performed by hand by an attentive graduate student (GS) with a pencil and a strong tolerance to boredom. Much of the complexity here lies in convincing the GS that it's a good use of time. Initial trials have not proved promising.

4.3 ARXIV INTEGRATION

We apply the SOAR scoring software implementation to the content of arXiv papers as a convenient Opera browser plugin. The effect of the plugin can be seen in Fig. 3: it provides a high-quality review of the work in question. Beyond the benefits of scalability, speed and consistency, this tool offers a direct "read/don't read" recommendation, thereby saving the reader valuable time which can otherwise be re-invested into rejecting reviewer invitations emails to compound its savings effect. We hope that this pre-review for pre-prints model will be of great utility to the research community.

5 CONCLUSION

In this work, we have introduced SOAR, a plug-and-play replacement for peer review. By striking an appropriate balance between pragmatism and our lofty goals, we anticipate near-instantaneous community adoption. In future work, we intend to further optimise our implementation of SOAR (from 2 LoC to potentially 1 or 0 LoC, in a ludic exercise of code golf). Other avenues of future research include peer-to-peer ego-limiting protocols and Tourette-optimal author feedback mechanisms.

REFERENCES


Maurizio Cattelan, a tar-covered seagull, and a very strange trip to the local 7-Eleven. Comedian. Art Basel Miami Beach, 2020. (presumably also visible while not under the influence of psychotropic substances).

Elliot J Crowley and Andrew Zisserman. The state of the art: Object retrieval in paintings using discriminative regions. 2014.


NYT-Photography. The 120,000bananawinsartbasel, 2019. URL. [Online; accessed 27-March-2020].


A APPENDIX

In the sections that follow, we provide additional details that were carefully omitted from the main paper.

A.1 TITLE PRONUNCIATION

In common with prior works, we hope that the arguments put forward in this paper will spark useful discussion amongst the community. Where appropriate, we encourage the reader to use the official title pronunciation guide in Fig. 4.

A.2 INHERITANCE OF PEERAGES

One historical challenge with the expansion of the UK peerage system has been something of a pre-occupation with preventing the passage of peerages to women. We note that this has grave implications for the ability to scale peer review. Consider the progressive rules for inheritance under Henry IV of England (Wikipedia contributors, 2020), which were as follows:
“If a man held a peerage, his son would succeed to it; if he had no children, his brother would succeed. If he had a single daughter, his son-in-law would inherit the family lands, and usually the same peerage; more complex cases were decided depending on circumstances. Customs changed with time; earldoms were the first to be hereditary, and three different rules can be traced for the case of an Earl who left no sons and several married daughters. In the 13th century, the husband of the eldest daughter inherited the earldom automatically; in the 15th century, the earldom reverted to the Crown, who might re-grant it (often to the eldest son-in-law); in the 17th century, it would not be inherited by anybody unless all but one of the daughters died and left no descendants, in which case the remaining daughter (or her heir) would inherit.”

Note that by avoiding the necessity of a direct bloodline between peers, SOAR neatly sidesteps this scalability concern, further underlining its viability as a practical alternative to traditional peer review.

A.3 New Insights: A Memoryless Model for Scientific Progress

Figure 5: By introducing a State-of-the-Art state transition diagram, we show how the progression of research can be modelled as a memoryless automaton.

Beyond time savings for reviewers, we note here that the SOAR score further provides insights into the scientific method itself, yielding time savings for authors too. To illustrate this, we provide a state transition diagram in Fig. 5 which models the evolution of research progress. Importantly, this model guarantees a Markov-optimal approach to research: a researcher must only ever read the paper which represents the current State-of-the-Art to make further progress.
A thorough investigation of the degree to which the COVID-19 pandemic has enabled subpar-quality papers to make it into the proceedings of SIGBOVIK, by reducing the supply of authors willing to invest the necessary effort to produce high-quality papers

Shalin Shah

Keywords: COVID-19, SIGBOVIK, lazy, quality, sadness, big Oof

Is this the longest Chess game?

Dr. Tom Murphy VII Ph.D.

Keywords: chess, chesses, chessing, G. K. Chesterton

Optimizing the SIGBOVIK 2018 speedrun

le060228

Keywords: speedrun, SIGBOVIK, pathfinding, DFS, Rust

Retraction of a boring follow-up paper to “Which ITG stepcharts are turniest?” titled, “Which ITG stepcharts are crossoveriest and/or footswitchiest?”

Ben Blum

Keywords: groove, in, retraction, the
A Thorough Investigation of the Degree to which the COVID-19 Pandemic has Enabled Subpar-Quality Papers to Make it into the Proceedings of SIGBOVIK, by Reducing the Supply of Authors Willing to Invest the Necessary Effort to Produce High-Quality Papers

Shalin Shah
Carnegie Mellon University
April 1, 2020

Abstract:
Based on the inclusion of this paper in the proceedings of SIGBOVIK 2020, we find that the COVID-19 pandemic has in fact enabled subpar-quality papers to make their way into the proceedings of SIGBOVIK, through a drastic reduction in the supply of authors willing to invest the necessary effort to produce high-quality papers.

Introduction:
Y’all know what COVID-19 is.

Methods and Materials:
You’re looking at the materials. Note that, in order to emphasize the subpar quality of this paper, we have opted to use extremely lazy Microsoft-Word default formatting, rather than LaTeX. Also, we have restricted the contents of this paper to a single page, to highlight its lack of substance. Meanwhile, our method was to simply submit this paper to SIGBOVIK 2020 and see what happened.

Results:
As evidenced by the fact that you’re currently reading this in the SIGBOVIK 2020 proceedings, this paper successfully made it into the SIGBOVIK 2020 proceedings.

Discussion:
The results indicate that SIGBOVIK 2020’s standards of quality have indeed fallen significantly relative to 2019, presumably due to the COVID-19 pandemic decreasing the supply of authors willing to invest the necessary effort to produce high-quality paper submissions.

Conclusions:
In conclusion, COVID-19 sucks.

References:
n/a
Paper 36: A thorough investigation of the degree to which the COVID-19 pandemic has enabled subpar-quality papers to make it into the proceedings of SIGBOVIK, by reducing the supply of authors willing to invest the necessary effort to produce high-quality papers.

Reviewer: Exponentiator
Rating: Better and better every day, by leaps and bounds and leaping leaps and bounding bounds
Confidence: Those guys have faith, why shouldn’t I?
1 Introduction

In my experience, most chess games end in a few moves. If you want to play a lot of chess moves, you just play a lot of chess games. Still, there are games that seem to go on forever. Perhaps the players are trying to lull each other into a false sense of security while waiting for the moment to strike, or perhaps they are stalling in a game of Chess to the Death.

Although many people “know how to play” chess, almost nobody fully understands the rules of chess, most authoritatively given by FIDE [1]. (See for example Figure 1 for a minor chess scandal that erupted in 2019 over an obscure corner case in the rules.) Several of these “deep cuts” have to do with game-ending conditions that were introduced to avoid interminable games.

Many chess moves are reversible (e.g. moving the knight forward and back to its starting position [4]), so informal games of chess could last forever with the players repeating a short cycle. In AD 1561, Ruy López added the “fifty-move rule” to prevent infinite games. This rule (detailed below) ensures that irreversible moves are regularly played, and so the game always makes progress towards an end state. Another rule, “threefold repetition” also guarantees termination as a sort of backup plan (either of these rules would suffice on its own).

So, chess is formally a finite game. This is good for computer scientists, since it means that chess has a trivial O(1) optimal solution. This allows us to move onto other important questions, like: What is the longest chess game? In this paper I show how to compute such a game, and then gratuitously present all of its 17,697 moves. Even if you are a chess expert (“chexpert”), I bet you will be surprised at some of the corner cases in the rules that are involved.

Speaking of rules, let’s first detail the three main rules that limit the length of the game. These rules cause the game to end in a draw (tie) when certain conditions are met.

1.1 The seventy-five move rule

The “fifty-move rule,” [8] as it is usually known, requires that an irreversible move is played at a minimum pace. For the sake of this rule, irreversible moves are considered captures and pawn moves, including promotion and en passant (which is also a capturing move) but not the movement of a previously promoted piece. Specifically [1]:

9.3. The game is drawn, upon a correct claim by the player having the move, if:

a. he writes his move on his scoresheet and declares to the arbiter his intention to make this move, which shall result in the last 50 moves having been made by each player without the movement of any pawn and without any capture, or

b. the last 50 consecutive moves have been made by each player without the movement of any pawn and without any capture.

Note that what is provided here is the option for a player to claim a draw (and the two provisions essentially allow either player to claim the draw at the moment of the 50th move). If neither player is interested in a draw, either because they think their position is still winning, or are just trying to create the longest ever chess game, the game legally continues. That’s why what is actually relevant for this paper is provision 9.6, which defines a draw:

![Fischer Random chessboard](image_url)

Figure 1: (Nepomniachtchi – So, 2019.) White to move during a speed Chess960 (aka “Fischer Random”) tournament. In this variant, the pieces start in different positions, but casting rules are such that the king and rook end up on the same squares that they would in normal chess. As a result, it is possible for the king or rook to not move during castling, or for the destination square for the king to already be occupied by the rook. Attempting to castle in the position depicted, grandmaster Ian Nepomniachtchi first touched the rook to move it out of the way. However, piece-touching rules require that when castling, the player first moves the king (and “Each move must be played with one hand only”); but how? The rook is occupying g1! One commenter suggested tossing the king into the air, then sliding the rook to f1 while the king is airborne, and then watching the king land dead center on its target. The arbiter required Nepo to make a rook move instead, but this was later appealed, and the game replayed.

© 2020 the Regents of the Wikiplia Foundation. Appears in SIGBOVIK 2020 with the ShortMoveString of the Association for Computational Eeesy; IEEEEEE! press, Verlag-Verlag volume no. 0x40-2A. c17,697
9.6. If one or both of the following occur(s) then the game is drawn:

... 9.6.2. any series of at least 75 moves have been made by each player without the movement of any pawn and without any capture. If the last move resulted in checkmate, that shall take precedence.

Draws after 75 moves (per player, so really 150 moves) are compulsory.

Interestingly (at least as interesting as anything in this dubious affair) it is known that some otherwise winning endgame positions require more than 50 moves to execute (Figure 2). The rules of chess have at various times allowed for longer timers in such known situations, but were later simplified to the fixed 50 (and 75) move limit.

1.2 Fivefold repetition

The 75-move rule is rarely applied in practice, but its counterpart, “threefold repetition” [9] is often the cause of draws in modern chess. This rule states that if the same position appears three times, the players can claim a draw:

9.2.2. Positions are considered the same if and only if the same player has the move, pieces of the same kind and colour occupy the same squares and the possible moves of all the pieces of both players are the same. [...] Like the 75-move rule, this rule has an optional version (upon three repetitions) and a mandatory one in 9.6:

[The game is a draw if ...]
9.6.1. the same position has appeared, as in 9.2.2, at least five times.

Fascinatingly (at least as fascinating as anything in this questionable undertaking), this rule used to require consecutive repetition of moves. However, there exist infinite sequences of moves with no consecutive n-fold repetition. For example, in the starting position, white and black can move either of their knights out and back. Let 0 be \( \mathcal{C}c3 \mathcal{A}c6 \mathcal{C}b1 \mathcal{A}b8 \) (queenside knights move, returning to the starting position) and 1 be \( \mathcal{C}f3 \mathcal{A}f6 \mathcal{C}g1 \mathcal{A}g8 \) (kingside). Now the Prouhet–Thue–Morse sequence [7] 0110100110010110... can be executed. This infinite sequence is cube-free (does not contain SSS for any non-empty string S) [6], and therefore never violates the consecutive threefold repetition rule [2].

Many implementations of chess ignore these rules or treat them incorrectly. Implementation of the seventy-five move rule simply requires a count of how many moves have transpired since a pawn move or capture, but programs typically do not force a draw after 75 moves. Repetition requires more work, since the program must keep track of each of the states reached since the last irreversible move. There are also some corner cases, such as ambiguity as to whether the starting position has “appeared” before the first move [4]. The ubiquitous FEN notation for describing chess positions does not even include any information about states previously reached.

\[ s_0 \text{ be the string } *0* \text{, then } s_i = s_{i-1}s_{i-1} \text{ where } \overline{0} = 1 \text{ and } \overline{1} = 0. \]

This clearly includes the well-known material-based cases like king and knight vs. king, but it also surprisingly includes many other specific positions, especially those with forced captures (Figure 3).

1.3 Dead position

The informal version of this rule (“insufficient material”) states that if neither side has enough pieces to mate the opponent (for example, a king and bishop can never mate a bare king) then the game is drawn. Again, the formal rule is more subtle:

5.2.2. The game is drawn when a position has arisen in which neither player can checkmate the opponent’s king with any series of legal moves. The game is said to end in a ‘dead position’. This immediately ends the game...

This game clearly includes the well-known material-based cases like king and knight vs. king, but it also surprisingly includes many other specific positions, especially those with forced captures (Figure 3).

\[ a b c d e f g h \]

Figure 2: Black to move and mate in 545 moves (!). The position was found (by Zakharov and Makhnichev [10]) while building an endgame tablebase of all possible 7-piece positions. Of course, the game ends prematurely in a draw because of the 75-move rule.

\[ a b c d e f g h \]

Figure 3: Black to move and draw in 0 (!). Most players and even chess experts believe that the only legal move is Kxna2, and that the game then ends in a draw with “insufficient material.” However, this game is already over. Since neither black nor white can win via any series of legal moves, by rule 5.2.2 the game immediately ends in a ‘dead position’. (Although see Section 1.3.1 for possible ambiguity in this rule.)
The insufficient material rule is curious in that it requires non-trivial computation to implement. In order to know whether the position is a draw, an implementation needs to be able to decide whether or not a series of moves that results in checkmate exists. Note that this is not nearly as bad as normal game tree search because the two sides can collaborate to produce the mate (it is not a ∃∀∃... but rather ∃∃∃...). Still, such “helpmates” can still be quite deep (dozens of moves) and are interesting enough to be a common source of chess puzzles. Proving the non-existence of a helpmate can be very difficult indeed (Figure 4).\(^4\)

![Figure 4: Thinking of implementing the rules of chess? To be correct, you’ll need your program to be able to deduce that no helpmates are possible in this position and thus the game is over.](image)

**Figure 4:** Thinking of implementing the rules of chess? To be correct, you’ll need your program to be able to deduce that no helpmates are possible in this position and thus the game is over. Stockfish rates this as +0.4 for white, even searching to depth 92. (Position is due to user supercat on Chess StackExchange.)

### 1.3.1 Ambiguity

Moreover, this rule contains some ambiguity. The phrase “any series of legal moves” is usually taken to mean something like, “the players alternate legal moves and follow most of the normal rules of chess.” In my opinion it is hard to justify an interpretation like this.

First, the rules specifically define “legal move”, with 3.10.1 saying “A move is legal when all the relevant requirements of Articles 3.1 – 3.9 have been fulfilled.” These requirements describe the movement of each piece as you are familiar (e.g. 3.3 “The rook may move to any square along the file or the rank on which it stands.”). They also disallow capturing one’s own pieces, or moving when in check. However, they allow as legal some moves that would otherwise be prohibited, like capturing the opponent’s king (this is excluded by 1.4.1, outside the definition of “legal”). Capturing the opponent’s king is generally not useful for demonstrating that checkmate is possible (Figure 5), so this is mostly a curiosity.

Second, what is a “series” of legal moves? It seems completely consistent to allow the white player to make several legal moves in a row, for example. The rules about alternating moves are again outside the definition of “legal move” and “series” is never defined.

We could instead interpret “any series of legal moves” as “taking the entirety of the rules of chess, any continuation of the game that ends in checkmate for either player.” I like this better, although it creates its own subtle issues. For example, should the position be considered dead if there is a checkmating sequence, but it requires entering a fivefold repetition or exhaustion of the 75-move rule? If so, this would end the game prematurely, and so it has implications for the longest possible game (Section 2.1). Even more esoterically, this interpretation causes the rule to be self-referential: A sequence must also be allowed by the rule being defined. A normal person would take the “least fixed point” (in the Kripke sense [3]) of this self-referential definition (fewest positions are drawn). But it is also consistent to interpret it maximally—in which case the longest chess game is zero moves!

For completeness, note that there are other routes to a draw (stalemate, draw offers) which we can ignore; it is easy to avoid these situations when generating the longest game.

### 2 Generating the longest game

It is generally not hard to avoid repeating positions, so the main obstacle we’ll face is the 75-move rule. Let’s call an irreversible move that resets the 75-move counter a critical move; this is a pawn move or capturing move (or both). The structure of the game will be a series of critical moves (I will call these “critical” moves) with a maximal sequence of pointless reversible moves in between them. If we execute the maximum number of critical moves and make 149 moves (just shy of triggering the compulsory version of the 75-move rule) between them, then this will be fairly easily seen as a maximal game.

In fact, for most positions, it is easy to waste 149 moves and return to the exact same position. So, the strategy for generating the longest game can mostly be broken into two tasks: Make a game with a maximum number of critical moves (it can also contain other moves) and then pad that game out to maximum-length excursions in between its critical moves.\(^5\)

---

\(^4\)Perhaps an enterprising reader can prove that for generalized chess, it is NP-hard to decide whether the game has ended due to this rule?

\(^5\)The first phase was constructed by hand. Software for inserting excursions and checking the result is available at: sf.net/p/tom7misc/svn/HEAD/tree/trunk/chess/longest.cc
2.1 Maximal critical moves

Critical moves are pawn moves and captures. There are 16 pawns, each of which has 6 squares to move into before promoting, so this is $16 \times 6$ critical moves. There are 14 capturable pieces, plus the 16 pawns (they can be captured after promoting); capturing them nets an additional $16 + 14$ critical moves for a total of 126. Each critical move can be made after a maximum of $75 + 74$ reversible moves, giving $(75 + 74 + 1) \times 126 = 18900$ moves. The final move would capture the last piece, yielding a draw due to the remaining kings being insufficient material (“dead position”). This is our starting upper bound.

We will not quite be able to use the entire critical move budget. If pawns only move forward in single steps, they will eventually get blocked by the opposing pawn on the same file. Pawns can move diagonally off their starting file only by capturing. We have plenty of capturing to do anyway, so this is no problem. With four captures per side, the pawns can be doubled, with a clear route to promotion, like in Figure 6.

![Figure 6: One way to clear the promotion routes for all pawns with eight captures.](image)

However, each of these captures is both a pawn move and a capturing move. This means that we lose $4 + 4$ critical moves off our total budget. $150 \times (126 - 8) = 17700$ is the new upper bound.

Parity. If a critical move (such as a pawn move) is made by white, then the first non-critical move is made by black. The players alternate these pointless moves until black has made 75 and white 74. Now it is white’s move, and white must make a critical move or the game ends due to the 75-move rule. If instead we wanted black to make the next critical move, this would happen after black has made 74 and white 74 non-critical moves. Any time this happens we lose one move against the upper bound. So, we want to minimize the number of times we switch which player is making the critical moves. Obviously we must switch at least once, because both black and white must make critical moves. This reduces the upper bound to 17699.

Starting condition. The first critical move should be made by black. The starting position (with white to move) is analogous to the situation just described, as if black has just made a critical move, and it’s white’s turn. White will play 75 reversible moves, black plays 74, and the 150\textsuperscript{th} move is black making the first critical move.

Note that white is quite constrained during this beginning phase, as pawn moves are critical moves and must be avoided. Only the white knights can escape the back rank. When we try to insert 149 pointless moves, we’ll only be able to move the knights and rooks, and doing this 75 times must leave e.g. one of the knights on an opposite-colored square.\textsuperscript{9} So we have to be a little careful about the position in which we make black’s first critical move.

Since white can only free their two knights, these are the only pieces that can be captured by black pawns. So it will not be possible for black to double four sets of their pawns as in Figure 6. This would require white to have a phase of critical moves to free pieces to capture, then black again to finish doubling pawns, then white again to promote its freed pawns. Each switch costs one move off the na"ive max. We can be more efficient with an asymmetric approach.

Black’s first phase of critical moves instead results in this:

![The white b and g pawns have a clear route to promotion. White can free each of the remaining 6 pawns with a single capture. Between this and black’s two pawn captures, this is the optimal 8 pawn moves that are also captures. Since black has plenty of freed pieces, white can promote all of their pawns during their own phase of critical moves, resulting in:](image)

Now black can promote all of its pawns and capture white’s pieces. We actually leave the white queen; this turns out to be essential:

\textsuperscript{9}Knight moves always change the color of the knight’s square, and same for rook moves constrained to the a1/b1 or g1/h1 squares.
Finally, white captures all of black’s pieces, and mates the black king:

Since we switch which color is making critical moves a total of three times, we must come shy of the naïve maximum of 17,700 moves by three. This gives us an upper bound of 17,697 for this approach, which we will be able to achieve.

**Ending condition.** The way this game ends is subtle for several reasons. First, note that we left the white queen on the board and used it for mate. It is required that white be the one mating for parity reasons, similar to the reason black must make the first critical move. Since white makes critical moves in the last phase, black leads on non-critical moves; at the moment white makes the checkmating move, black has made 75 non-critical moves, and white 74. White’s 75th move mates.

But doesn’t this trigger the 75-move rule? No, this rule (9.6.2; Section 1.1) has a special exception for checkmate: “If the last move resulted in checkmate, that shall take precedence.” Essentially, we can treat checkmate as a type of critical move.

Why not capture the white queen too? This would be a critical move, but once two kings remain, the game ends immediately in a draw. So there is no length advantage here over checkmate. In fact, attempting to capture may shorten the game: Consider the position in Figure 3 where black is forced to capture the queen. This game is over prior to the capture, so this is shorter than the checkmating sequence! Even if the king had an escape square, it is arguable (Section 1.3.1) that the game is over in any case: There is no “series of legal moves” that leads to mate. Either the king captures the queen (and then clearly no mate is possible with just the two kings) or the king escapes, but in doing so plays the 75th non-critical move, and triggers a draw by the 75-move rule.

The most foolproof way to ensure that mate is always possible is for the game itself to end in mate. This sidesteps any ambiguity about the way the 75-move condition should be interpreted, as well.

### 2.2 Inserting excursions

The game described in Section 2.1 has 118 critical moves in 289 total moves. There is some inefficiency between the critical moves, but this doesn’t matter since we are trying to generate a long game anyway. In fact, the next step will be to add as much inefficiency as possible in between the critical moves.

Each “critical section,” which is the series of moves ending in a critical move, can be treated independently. If black ends the section with a critical move, then we want $75 + 74$ non-critical moves to be played, and then black’s critical move. For white, of course, $74 + 75$. There are many ways we could try to make these 149 moves; we don’t even have to use the moves that are already there as long as we end up in the right position to make the critical move. But a simple approach suffices.

For each critical section, we loop over all of the positions encountered, and attempt inserting excursions that return us to the same position but waste moves. There are two types of excursions we try: Even excursions (each player makes two moves) and odd (each player makes three moves).

**Even excursions.** This four move sequence moves two pieces of $X_1$ and $X_2$ of opposite colors. $X_1$ moves from $s_1$ to $d_1$ then $X_2$ from $s_2$ to $d_2$, then $X_1$ moves from $d_1$ back to $s_1$, and $X_2$ from $d_2$ back to $s_2$. Easy. Any piece can perform this maneuver other than pawns (which would be critical moves anyway) as long as there are legal squares (considering check, etc.). All of $s_1, d_1, s_2, d_2$ must be distinct. No shorter excursions are possible.

**Odd excursions.** This is the straightforward extension to three squares ($s_1 \rightarrow m_1 \rightarrow d_1 \rightarrow s_1$), for a six-move sequence. The squares for each piece must be distinct but it is possible for e.g. $m_2$ to equal $s_1$. Knights cannot perform this trick: each move changes the color of the square the knight sits on, which causes a contradiction with a cycle of length three. All other pieces can do it with sufficient room. The king, for example, can move horizontally, then diagonally, then vertically back to its starting square.

Note that odd excursions are not possible early in the game (prior to white moving any pawns), because even when the knights are free, the rooks only have two squares (and thus the same color parity argument applies as knights). Some opportunity can be created by having a black knight capture one of white’s bishops. Fortunately, we do not need any odd excursions at this point in the game.

We find excursions by just looping over possible moves that satisfy the criteria, prioritizing odd excursions if the target (divided by two) is odd. In order to avoid triggering the fivefold repetition rule, we also keep track of all of the positions encountered, and never enter a position more than two times. (Here we avoid even threefold repetition.) It is not necessary to look beyond the critical section, because critical moves make it impossible to return to a prior position.

This process is not at all guaranteed to work; it may fail to fill the critical section. Indeed, as discussed in Section 2.1, we must have at least one move of slack whenever we switch from a critical move by one player to the other. In practice this approach succeeds readily, and manages to waste 149 moves in each critical section of the input game, save for the three times that parity requires one move of slack. The full game is uselessly included in the very tiny font in Section 4.
3 Reader’s guide

The paper demonstrates a game with three “switches” of which side is making critical moves; each costs a move against the naïve maximum due to parity. We clearly need at least one switch (both sides must make critical moves), but is it possible to do it with only two? If not, can this lower bound be proved?

The game given is believed to be maximal, as measured in the number of moves. But, other metrics exist. For example, the letter $g$ is slightly wider than $f$, so moving $\text{g}3$ is typographically longer than $\text{f}1$. PGF format itself can be stretched by making moves that need to be disambiguated (\textit{f7f7} means the move the \textit{f} on the f file to f7? (not “the giant sword from Final Fantasy 7” as many believe)) or checking the opponent’s king for a bonus +.

Some moves are in terms of distance traveled; moving the queen or bishop opposite corners is $\sqrt{2} \times 7$ squares! What is the longest game according to these or other metrics?

In Chess, it is impossible to capture your own pieces. How does this limitation apply to your own life?

In many games of Chess, the black and white pieces are found to disagree. What does this say about society?

How do you feel about the ending of the game? Is it disappointing that the result is not symmetric (i.e. a draw), or elegant that it demonstrates its own avoidance of the “dead position” rule? Was it what you expected?

The character called $\text{W}$ often plays an important role in the story. How would you describe her/his personality? How does she develop over the course of the game?

Is this the longest Chess game? is the author’s sixth paper about chess for SIGBOVIK, but it is widely believed that nobody wants to read this kind of thing. What is wrong with him?

4 A longest game

There are jillions of possible games that satisfy the description above and reach 17,697 moves; here is one of them. Critical moves are marked with [bold]. Note that in the standard PGF format for listing games, move numbers are “full moves” consisting of a move by white and then black, whereas we use “move” in this paper to mean “half-move”: an individual move by one of the players. Thus the game ends during full move 8,849, one half of 17,697.

It can also be downloaded at tom7.org/chess/longest.pgn. Many chess programs fail to load the whole game, but this is because they decided not to implement the full glory of chess.
Optimizing the SIGBOVIK 2018 speedrun

leo60228
@leo60228
April 1, 2020

Abstract

SIGBOVIK 2018 [1] is a popular game. However, very little work has been put into optimizing its speedrun. In this paper, I will demonstrate usage of pathfinding algorithms for the goal of making the optimal speedrun of SIGBOVIK.

1 Motivation

I was bored.

2 Method

I first looked at implementations of pathfinding algorithms in Rust. I initially wanted to use A* because it was popular. However, the API looked complicated. I found an implementation of DFS to use instead. This DFS implementation required me to define a start point, one or more end points, and a function to map from point to points. I implemented all three of these, using an enum of Start, Success, and Page, because I forgot that there was only one success page.

I used a depth first search algorithm in order to create the mapping. To do this, I clicked as far as possible along the first choices, adding to my mapping (a match block) as I went, then backtracking once I reached a cycle or the end. I then manually removed cycles because I wasn’t sure if the DFS implementation I was using supported them and I didn’t feel like writing my own.

After I had defined these three functions, I simply used the pathfinding [2] crate’s DFS function. This gave me an optimal path to program into LiveSplit One [3], a tool for speedrunning that I’ve seen speedrunners use, so I probably should too if I’m speedrunning. I opened it on my phone and added the order that my code gave me.

I recorded my phone and computer screen while I began the speedrun. I then combined the phone and computer screen and attempted to sync them from memory in a video editor. I then uploaded it to YouTube [4].
3 Results

My speedrun was 17.88 seconds, which I’m pretty sure is a world record among the SIGBOVIK 2018 speedrunning community. It can be viewed at https://youtu.be/VPrT8Y-aRhS. My code will be available on GitHub at https://github.com/leo60228/sigbovik2018 if I remember to make it non-private after SIGBOVIK.

4 Appendix: Code

```rust
use pathfinding::directed::dfs::dfs;

#[derive(Debug, PartialEq, Eq, Copy, Clone)]
pub enum Page {
    Start,
    Success,
    Page(isize),
}

macro_rules! pagevec {
    ({$($page:expr),*}) => {
        vec![$(Page::Page($page)),*]
    }
}

impl Page {
    pub fn successors(&self) -> Vec<Self> {
        match self {
            Page::Start => pagevec![47, 177, 205],
            Page::Page(47) => pagevec![153, 206],
            Page::Page(153) => pagevec![17],
            Page::Page(17) => pagevec![35, 135],
            Page::Page(35) => pagevec![51, 68],
            Page::Page(51) => pagevec![68], // cycle
            Page::Page(68) => vec![], // cycle
            Page::Page(135) => pagevec![124, 183],
            Page::Page(124) => pagevec![116, 88],
            Page::Page(116) => pagevec![208],
            Page::Page(208) => vec![Page::Success],
            Page::Page(88) => pagevec![208],
            Page::Page(183) => pagevec![207, 208],
            Page::Page(207) => vec![],
            Page::Page(206) => vec![],
            Page::Page(177) => pagevec![130, 117],
            Page::Page(130) => pagevec![154, 39],
        }
    }
```
Page::Page(154) => pagevec![17],
Page::Page(39) => vec![],
Page::Page(117) => pagevec![87, 50, 28],
Page::Page(87) => pagevec![69, 40, 28],
Page::Page(69) => vec![], // cycle
Page::Page(28) => vec![], // cycle
Page::Page(50) => vec![], // cycle
Page::Page(40) => pagevec![178], // cycle
Page::Page(178) => pagevec![208], // cycle
Page::Page(205) => vec![],
Page::Success => pagevec![],
_ => unimplemented!("{:?}", self),
}
}
}
fn main() {
println!("{:?}", dfs(Page::Start, Page::successors, |x| x == &Page::Success));
}

5

Appendix 2

I just realized I probably should have used more sections, so here’s one.

References
March 29, 2018.

34


Retraction of
a boring follow-up paper to
“Which ITG Stepcharts are Turniest?”
titled, “Which ITG Stepcharts are
Crossoveriest and/or Footswitchiest?”

Ben Blum
bblum@alumni.cmu.edu

In my 2017 paper, a boring follow-up paper to Which ITG Stepcharts are Turniest? titled, Which ITG Stepcharts are Crossoveriest and/or Footswitchiest? (Blum 2017), I wrote of maximum $T$, XO%, FS%, and JK% values as follows:

A chart could conceivably end right before such a step, sneaking through some small $\epsilon$ extra turniness (VII 2014) (similar to the case of 270s in (Blum 2016)),

[...]

By the way, the theoretical maxima for XO%, FS%, and JK% are 50-$\epsilon$, 100-$\epsilon$, and 100-$\epsilon$, respectively (VII 2014).

However, in the experimental results, Tachyon Epsilon (Matt 2013) placed among the lowest-ranking stepchart packs in every category, yet I neglected to properly cite Dr. VII’s landmark paper from 2014 at that time. Though we may never know what, if anything, is Epsilon? we now know at least what it is not: crossoveriest and/or footswitchiest.

In conclusion, please reject my paper. I messed up on it.

References
B. Blum. Which ITG stepcharts are turniest? SIGBOVIK, 2016.
B. Blum. Which ITG stepcharts are crossoveriest and/or footswitchiest? SIGBOVIK, 2017.
T. VII. What, if anything, is epsilon? SIGBOVIK, 2014.
We believe the subject of this paper is of utmost importance to the field, and appreciate its timeliness, timeline, timbre, and temerity. That being said, in order to be accepted into this prestigious venue, the we suggest the following changes:

- The main methodology is seriously flawed and must be addressed.
- The authors used the word “whilst” 11 times. We recommend a minimum “whilst” usage count of 20 to be seriously considered for publication.
- The English language writing in this paper contains numerous spelling and grammar errors. Consider kidnapping a native English speaker, chaining them in your basement, and depriving them of food until the draft is written in language that William Strunk would be proud of.
- Please use raster rather than vector images for all figures, so that the reader may count the pixels used in the each letter of the text in the axis labels.
- In order to give due credit to all contributors to this work, please make Reviewer 5/7 a co-author.
- On page 44, line 13, eight word, second character, we found the letter “o.” We like this letter, please leave it as is.
- Additional changes have been suggested in the attached PDF.
Programming Languages

7  Ntinuation copassing style  
Cameron Wong, Dez Reed  

  Keywords: continuation, ntinuation, control flow, programming languages

8  Formulating the syntactically simplest programming language  
Brendon Boldt  

  Keywords: programming languages, compilers, formal grammars

9  Type-directed decompilation of shell scripts  
Spencer Baugh, Dougal Pugson  

  Keywords: partial evaluation, type systems, shell scripting

10  Verified proof of P=NP in the Silence theorem prover language  
Matias Scharager  

  Keywords: soundness, P=NP, theorem prover, Silence, programming language theory, type theory
Abstract—We applied the categorical technique of “do that thing, but backwards” to continuation passing style to produce a novel programming idiom utilizing the COME FROM control flow operator.

I. INTRODUCTION

Continuations are an important abstraction in the analysis and description of functional programs. By manipulating them, a savvy programmer can express complex control flow precisely and unambiguously. Unfortunately, with great power comes great responsibility, and code written in continuation-passing style is often obtuse and unreadable to those without such arcane knowledge. It is even said that CPS is merely an obfuscation technique designed to terrify novice functional programmers and has no practical benefit.

Of course, obscure intermediate forms and programmer-facing implementations thereof need not be restricted to functional programs. A reasonable analogue may be the vaunted Static Single Assignment form, in which the programmer may need to use mystical ϕ-functions. As it turns out, however, SSA is actually equivalent to CPS and is therefore cheating. No, to achieve true parity, a different approach is needed.

Towards this end, we take a page from category theory, applying the time-worn approach of “flip everything backwards and see what happens”. Monads embed impurity to a pure language, so too can comonads embed (enforced) purity into an impure language, and thus “ntinuation copassing style”∗ is born.

A continuation is a typesafe abstraction of the “return address”. In other words, invoking a continuation is a type-safe delimited GOTO. A ntinuation, then, is the opposite of GOTO. For wisdom as to what, precisely, this entails, we turn to the programming language INTERCAL. While INTERCAL does have a form of GOTO, it also implements the COME FROM operator, which is precisely the opposite of GOTO!

What is copassing? The type of a CPS function in a traditional functional language typically involves (′a → ′b) → ′b, which has arrows. Entering into the true categorical spirit and flipping the arrows, then, suggests that “copassing” a ntinuation must be typed at (′a <- ′b) <- ′b. Unfortunately, this doesn’t mean anything, but it is fun to write and look at.

II. NTINUATIONS

In continuation passing style, the core observation is that, by passing the continuation as an argument, it allows the callee to manipulate it to construct new continuations and call them as necessary. In this way, the state of the overall program after any given subroutine invocation is neatly encapsulated into the continuation.

To flip this around, then, a ntinuation would be to increase expressiveness by removing control of not only the state of the program on subroutine return, but also the condition on which the return occurs.

A NCS-transformed function may contain no local variables, but instead relies entirely on its parameters and global, mutable state. The ntinuation, then, is subject to an exit condition imposed on the global state of the program. This process is known as “copassing” or “catching” the ntinuation. †.

This construction mimics the vaunted COME FROM control flow operator from the INTERCAL programming language in that a ntinuation represents a “trapdoor” that wrests control flow from the encapsulated routine on a given condition being met. In the true imperative spirit, ntinuations are given meaning entirely by global state and the manipulation thereof.

III. EXAMPLE

Here is the factorial function, written in Ntinuation Copassing Style pseudo-code extended with ntinuations and subroutines:

ROUTINE fact:

PLEASE NOTE that the input is passed in the
1: SET y TO y * x
SET x TO x-1
PLEASE COME FROM (1) WITH y WHEN x EQUALS (1)

∗Categorically-inclined readers may protest that we have not properly complemented the “style”. The authors believe that code written using COME FROM is already pretty much the opposite of “stylish”, and thus is already complementary.

†An initial draft of this work was titled “termination catching style”, which perhaps more neatly encapsulates this idea, but after much debate, we settled on “ntinuation” because it would be funnier watching people attempt to pronounce it.
In true INTERCAL fashion, the line beginning with PLEASE NOTE is a comment, as it begins with the text “PLEASE NOT”\(^\dagger\).

This is the standard iterative formulation of the factorial function that multiplies the global result variable \(y\) by a loop counter \(x\) until \(x\) reaches 0. The key difference, however, is in the wrapped recursive call to \texttt{fact} that declares a new continuation which aborts computation and returns to that point.

Note that, should multiple exit conditions apply at once, it is nondeterministic which trapdoor opens, as in the original formulation in INTERCAL. In this case, all trapdoors lead to the same place with the same condition, so they will take control one after the other in some nondeterministic order, but do nothing before yielding control to another trapdoor. Finally, after all trapdoors are opened, the routine halts with the value of \(y\) containing the result of the factorial.

**IV. Conclusion**

We have presented a novel control flow operator in the style of continuation passing style by attempting to “reverse the arrows”. In fact, the idea of conditional trapdoors (encapsulating \texttt{COME FROM}s) mimic that of pre-empting interrupts that take control from a process. In this way, just as CPS allows for precise manipulation of control flow by manipulating a subroutines’ next steps, NCS does the reverse by moving all control flow into global condition triggers, allowing programmers to focus on the individual steps of a given operation. We expect this tool to become key in an imperative obfuscator’s toolbox.

In future work, we hope to explore whether, as CPS forms a monad, NCS forms a true comonad (in fact, we believe that NCS also forms a monad, but we did not confirm this).

\(^\dagger\)The authors originally intended to give this example in an extended INTERCAL, but gave up after four hours of whiteboarding a multiplication algorithm.
FORMULATING THE SYNTACTICALLY SIMPLEST PROGRAMMING LANGUAGE

Brendon Boldt
Language Technologies Institute
Carnegie Mellon University

ABSTRACT
The syntax of a programming language is one of its most visible characteristics, and thus, heavily shapes how users both interact with and view the language. Abstruse syntax correlates to lower usability, and therefore, lower adoption of a programming language. In light of this, we present a programming language with the simplest possible syntax. In addition to the language specification we give a basic style guide and reference implementation.

Keywords programming languages · compilers · formal grammars

1 Introduction
As computers become further and further prevalent in modern day society, computer literacy similarly grows in importance. Taking the analogy further, composition is to literacy as computer programming is to computer literacy. A novice programmer will often find it difficult to address the syntax of a programming language at first, which hinders the acquisition and fluency with the semantics of the programming language [6]. In order to lessen this barrier of entry, we will explore a programming language with the simplest possible syntax.

The complexity of the syntax of a grammar is dependent on a number of well-defined factors: number of rules, cardinality of unique terminals (tokens), and number of terminals and non-terminals per production. These are all further characterized by probabilistic factors, namely their distributions and associated information-theoretic entropies. Determining the equivalence of two grammars is an undecidable problem, and the computation of the entropy of grammars and their syntax trees is an open area of research [7]. Thus, these considerations are beyond the scope of this paper. Regardless of the formal specifications of complexity and entropy, we assert that fewer unique tokens, fewer rules, and lower branching factors all correspond to simpler syntax.

The core concept of our programming language is that of simplicity. Fittingly, the name we have chose for the programming language is “SimPL” standing for “Simple Programming Language”. We present the formal specification of SimPL 2 in Section 2.

1.1 Related Work and Scope
SimPL bears some resemblance to so-called “esoteric” programming languages or “esolangs.” Such programming languages typically try to maximize goals, such as minimality and creativity, that typical programming languages typically leave by the wayside. Two well-known esolangs are brainfuck and Unlambda which are a Turing tarpit and Church quagmire respectively [2]. Such tarpits and quagmires seek to limit the semantics of a programming language to the bare minimum as a demonstration of how little is required to be computationally universal.

Another direction of programming language minimalism comes in the form of one instruction set computers (OISCs) [1]. These are specifically machine languages which provide only one instruction (operator) which, in turn, can take

---

Footnote 1
This is not to be confused “SymPL” or “Symbol Programming Language” which is the true name of APL.

Footnote 2
The astute reader will notice that we have skipped straight to SimPL 2 (typically, 1 comes before 2). For an explanation of this, see Footnote 1 (although 2 has preceded 1 in this case).
variable arguments. While OSICs and SimPL both center around the idea of “unity” in terms of representing a program in a language of some sort, SimPL does this at the level of a programming language while OSICs are, by definition, concerned with instruction sets.

While the aforementioned languages are mostly interested in testing the limits of minimalism in computability (either through limited semantics or instruction sets), our work adheres to area of language design purely with respect to syntax and syntactic usability. Thus, rather than introducing a language with entirely new semantics as well as syntax, we will recycle familiar semantics packaged in a novel syntactic model. In this sense, our language bears similarities to transpiled languages such as CoffeeScript or Dart.

2 Language Specification

2.1 Syntax

We begin with the Backus-Naur form of SimPL.

\[
\langle \text{start} \rangle ::= \langle \text{expr} \rangle
\]

\[
\langle \text{expr} \rangle ::= \langle \text{tok} \rangle \langle \text{expr} \rangle \mid \langle \text{empty} \rangle
\]

Note that there is only one lexical token SimPL, which can be represented an arbitrary character, emoji, or any other kind of thing. While we give the BNF of the grammar here. We have artfully crafted the syntax such that it does not need to be expressed as a context-free grammar (Type-2 on the Chomsky hierarchy). We can, in fact, express the syntax of SimPL with a finite state automaton (representing a Type-3 grammar). While this specification seems small, one could imagine it smaller, but we ran into serious problems attempting to use a simpler specification than presented above.\footnote{See Appendix A for specifications that are not SimPL.}

For example, a string can be determined to be within the grammar using the following regular expression (Perl compatible):

\[
.*
\]

Figure 1: Perl-compatible regex specification for SimPL.

2.2 Semantics

The semantics of a programming language are concretely expressed in machine code. This transformation is achieved through the compilation of a programming language. The relationship between the source code of different languages as well as their machine representation can be represented as a simple subcategory of $\text{Set}$. Namely, our objects will be the set of source strings for a given programming language with a special object for the set of all machine code representations. Morphisms from source code objects to machine code objects are realized by the process of compilation. If we take every programming language to have a canonical compiler implementation, assume that decompilers do not exist, and consider only one machine architecture, we can view the machine code object as the terminal object of our category.

We will use $S_C$ and $S_S$ to refer to the objects of the C and SimPL languages respectively and $M$ to refer to the terminal object corresponding to machine code representations. $k_C : S_C \to M$ and $k_S : S_S \to M$ are morphisms that correspond to the compilation of a programming language. While both GCC and Clang could be seen as different morphisms from $S_C$ to $M$ we would consider the languages to be distinct since the same source strings correspond to different machine code representations.

In this paper, we focus on simplicity of syntax, and in order to keep other factors constant, we tie the semantics to a well-established reference point, namely C. To express this more succinctly, we will introduce the term semantic equivalent in C or SEC. A SimPL source code string corresponds (semantically) to a unique C source code string—this C source code is the SEC of the SimPL source code. We formally define the semantics of SimPL as $k_S = k_C \circ r_S$ where $r_S$ is one component of an isomorphism between semantically equivalent C and SimPL source code. The components of this isomorphism are $r_C : S_C \to S_S$ and $r_S : S_S \to S_C$ such that $r_C \circ r_S = \text{id}_{S_S}$ and $r_S \circ r_C = \text{id}_{S_C}$. This is the illustrated by the commutative shown in Figure 2.

In this paper, we focus on simplicity of syntax, and in order to keep other factors constant, we tie the semantics to a well-established reference point, namely C. To express this more succinctly, we will introduce the term semantic equivalent in C or SEC. A SimPL source code string corresponds (semantically) to a unique C source code string—this C source code is the SEC of the SimPL source code. We formally define the semantics of SimPL as $k_S = k_C \circ r_S$ where $r_S$ is one component of an isomorphism between semantically equivalent C and SimPL source code. The components of this isomorphism are $r_C : S_C \to S_S$ and $r_S : S_S \to S_C$ such that $r_C \circ r_S = \text{id}_{S_S}$ and $r_S \circ r_C = \text{id}_{S_C}$. This is the illustrated by the commutative shown in Figure 2.

The expressivity of the syntax of C an SimPL may seem too widely disparate to be practical, yet we can actually define the isomorphism between C and SimPL source code. In particular, we will call this isomorphism radix representation.
mutation (RRM). As C can be represented as a sequence of ASCII-encoded bytes, we can express any C program as a radix 128 number (as standard ASCII values can be represented with 7 bits) with each numerical place corresponding to an ASCII character in the C program. SimPL, on the other hand, is represented by a radix 1 number. Thus, we can morph C source code into SimPL as follows.

$$\sum_{i=0}^{n-1} c_i \cdot 128^i = N \rightarrow 0_00_10_2\ldots0_{N-1} \quad (1)$$

Similarly, given a SimPL program of length $N$, we can generate the SEC where the zero-indexed $i$th character corresponds to:

$$\left\lfloor \frac{N}{128^i} \right\rfloor \mod 128. \quad (2)$$

2.3 Syntax vs. Semantics

It is worth addressing briefly a common objection to our delineation between syntax and semantics. The objection is that SimPL’s syntax is simply a trivial shell and that the real syntax (that of C) is masquerading as the “semantics” of SimPL. Hence, the true syntax of SimPL is just as complicated as that of C. One illustration of the objection consists in the fact that SimPL has no “syntax errors” per se but instead mutates C syntax errors into SimPL “semantic errors.”

In response, we first point out that there are many different levels of errors beyond syntax and semantic ones. In roughly ascending level of “depth” we have: lexical (errors), syntax, semantic, logical, and design-level errors (though this list is not exhaustive). Although these levels are not strictly defined, one of the sounder heuristics for determining the level of an error comes from looking at from which stage of the compiler the error comes. The fact is that syntax errors in the corresponding SEC would be generated by the compiler backend after parsing and AST generation, and thus would be out of the scope of true “syntax errors.”

Furthermore, in some cases there is not even a clear distinction syntactical and semantic errors. For example, in Python, using return or yield outside of a function yields a SyntaxError. Although trying to return or yield from outside of a function is a semantic error rather than a syntactical one, this difference is not maintained in Python [3]. Further debate as to the precise distinction between syntax and semantics could provide useful topics for future work but is beyond the scope of this paper.

3 Representation

3.1 Naïve Representation

The naïve representation of SimPL entails representing each token as a single character. For example, representing the SEC $A$ in SimPL could take the various forms specified in Figure 3. It becomes evident when we give the naïve representation of the SEC $AB$ that this mode of representation quickly becomes unwieldy; for formatting purposes, the SimPL source string has been rendered in Appendix B.

As is the case with all programming languages, syntactically correct does not imply readable or maintainable. Thus, we present some potential stylistic improvements in order to reduce cognitive load when determining the number of
characters in Figure 4. The choice of using delimiters every 10 is somewhat arbitrary. This is certainly appropriate for situations where the program length is not many orders of magnitude greater than 1 but could be counter-productive otherwise. Thus, we could turn to constant multiplicative spacing between delimiters instead of constant additive spacing.

While these intermediary delimiters can make it easier to track one’s place in the source code, generating such delimiters goes beyond what is strictly necessary. For example, we can simply express the number of tokens at the end of the program.

### 3.2 Practicality

In a more practical example, take the basic SEC of a “Hello, arXiv!” program shown in Figure 5. The above program consists of 86 characters; thus, the naïve representation consists of \((2^7)^{86} = 2^{602}\) tokens or about \(2^{699}\) to \(2^{699}\) bytes (if each token is represented by 1 byte). More generally, an \(n\)-character SEC requires \((2^7)^n\) bytes to represent.

At very large-scale code bases, simply comprehending the scale of the representation (let alone actually representing) becomes difficult. Take the Linux kernel, it has on the order of \(2 \times 10^7\) lines of code [5]. If we use the estimate of an average of 40 characters per line, that gives us \(8 \times 10^8\) characters. The SimPL representation of this would, then, require on the order of \((2^7)^{8 \times 10^8} = 2^{610}\) tokens to represent. Any explicit representation of this number of tokens surpasses any current or theoretically possible information system. Although this exhibits exponential growth, this number is not very large in the context of mathematics, being \(2^{10^d} < n_{\text{Linux}} \ll 2^{10^c}\), but this number is still large enough to make pursuing more efficient modes of representation prudent. Thus, while this mode of representation works well for explaining the conceptual grounding of SimPL, a more efficient mode of representation is needed to effectively store and process SimPL.

### 3.3 Compressed Representations

Let us revisit Figure 4e, namely the representation which consists of a repeated, non-numeric character followed by the number of total tokens at the end (expressed as a radix 10 number). By observing this style of representation, we can see that number at the end by itself (i.e., without the repeated leading characters) could serve as a sort of shorthand representation of the whole program. This sort of shorthand makes the task of representation far more tractable. For example, simply using a radix 10 representation for the “Hello, arXiv!” SEC would as presented above would yield \(\lceil \log_{10} 2^{602} \rceil = 182\) characters in total.

Using a higher radix could give us an even more compact representation; for example radix 64, common in text-based data transmission would give us \(\lceil \log_{64} 2^{602} \rceil = 101\). If we take this even further, we could use the cardinality of Unicode 12.1 characters which stands at 137,994, which leads us to an even more compact representation \(\lceil \log_{137,994} 2^{602} \rceil = 36\). Although, this becomes unwieldy not by virtue of its length but on account needing to have complete familiarity with
#include <stdio.h>

int main() {
    printf("%s\n", "Hello, arXiv!");
    return 0;
}

Figure 6: The Canonical SimPL of the SEC shown in Figure 5.

every Unicode character. Thus, the optimal representation will strike a balance between compactness and recognizability of the set of characters used.

“Canonical SimPL,” as we call it, uses a radix 128 representation of an SEC where each digit is rendered as its ASCII equivalent (e.g., 65 as A). An immediate issue seems to arise from the fact that there are 33 unusable values (either non-printable characters or unmapped values) in the ASCII encoding [4]. Yet due to RMM with C, any semantically valid Canonical SimPL source code will only ever correspond to printable ASCII characters. In this way, just as RMM provides us with an isomorphism at the machine-interpretable semantic level, Canonical SimPL provides us with a sort of isomorphism at the human-interpretable level. As an illustration, we have shown the same “Hello, arXiv!” program written in Canonical SimPL in Figure 6.

4 Reference Implementation

In our reference implementation of SimPL, we limited our scope to compiling Canonical SimPL source code. This compilation, in fact, can be done entirely with a typical *nix toolchain. For example, for any given Canonical SimPL program lorem.spl, the compiled binary can be generated as such:

```
cp lorem.spl lorem.c && gcc lorem.c -o lorem
```

This represents the GCC dialect of Canonical SimPL; the Clang dialect would be implemented similarly.

Acknowledgements

Special thanks Dr. Bartosz Milewski for informative YouTube lectures on category theory.

References

[2] In Wiki. TuringTarpit

A Earlier SimPL Iterations

SimPL 0 was described by the empty grammar. Though the syntax was very simple, the semantics of the program were static, that is, SimPL 0 could describe one and only one program. We found these semantics to be unreasonably limiting. Thus, we decided that a non-empty grammar would be necessary.
Simple 1.0 introduced having at least one rule in the grammar. Although it now conceivable that program could be written according to the grammar (namely a single token), there is still only one possible program. In this way, SimPL 1.0 is equivalent to SimPL 0. We proceeded to Simple 1.1 as such:

\[
\text{start} \ ::= \langle \text{expr} \rangle
\]

Although programs can now consist of multiple tokens, the only valid program is in fact one consisting of an infinite string of tokens. In this way, SimPL 1.1 is equivalent to SimPL 1.0. Finally, we added the grammatical option to terminate a multi-token program which we present as SimPL 2 as described in the body of the paper. In short, SimPL 0, 1.0, and 1.1 are too simple but 2 SimPL is not.
Reviewer: Wannabe  
Rating: Potential breakthrough  
Confidence: I couldn’t make heads or tails of it, but something this complex must surely be terribly clever

Insightful framing of a classic problem in a completely novel way
Type-directed decompilation of shell scripts

Spencer Baugh
University of Carcosa
Carcaso, Hyades
first.last@gmail.com

Dougal Pugson
Pugson’s C++ Crypt LLC
New York, USA
dougalphugson@gmail.com

Abstract
Maintenance shell programmers are often faced with inscrutable shell scripts without human-readable source code. We apply techniques pioneered by the type-directed partial evaluation community to create a decompiler which can take an executable shell script and recover its original source code. This technique has surprising generality, and our decompiler can also be used as a pretty-printer, or in general, as a compiler from any language into shell.

CCS Concepts · Software and its engineering → Scripting languages; Compilers; Software maintenance tools; · Theory of computation → Type theory.

Keywords · Partial evaluation, type systems, shell scripting

ACM Reference Format:

1 Introduction
Among its many beneficial properties, shell has a unique feature: The language automatically compiles itself as it is written. Immediately after a shell script is typed into a computer and saved into a file, it is transformed into a compiled form which is unreadable by humans. ¹ Such a script can be immediately used for all your most important financial data, medical procedures, etc., while you are safe in the knowledge that no-one can read your important proprietary shell scripts.

Unfortunately, sometimes such scripts might exhibit minor bugs. Since these executable files that are leftover from the process of shell scripting are unreadable garbage, maintenance programmers are often forced to rewrite from scratch. Sometimes, very brave programmers will try open one of these old, buggy "shell script files", but generally they are instantly turned to stone upon seeing the inscrutable contents. And since shell is so convenient, typically the original programmer is long gone - probably retired early.

Our contribution in this paper is a decompiler for these files, which is able to recover the original source code. This source code can then be viewed by the maintenance programmer, to help them reimplement the script in their non-shell language of choice.

As outlined in our previous work, [1], the Unix shell is closely historically related to functional programming. Thus it should be no surprise that we are able to transplant techniques from the functional programming community, to the shell, in the tradition of [4].

In this case, we use partial evaluation techniques to achieve our goal. As described in [3], as citing [2], as blowing the minds of undergrads everywhere, sufficiently general partial evaluation techniques can be applied to "reify" a compiled program and recover the program source code, under the constraints that the program 1. has a type and 2. terminates. Pretty easy constraints, I think we can make that happen for bash!

As a more approachable introduction (note that "more approachable" doesn’t mean "approachable"), these techniques can be compared to tagless-final-style techniques. Decompilation of a typed, compiled program is essentially identical to pretty-printing of a tagless-final-style term. The magic is that any typed, closed program can be treated as a TFS term.

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 contains a exploration of the requirements for applying these techniques, and establishes their firm grounding in theory. Section 3 demonstrates several applications. Section 4 gives an overview of the implementation. Section 5 concludes the paper, discusses future work, and affirms that this was the right thing to do.

2 Theory
As mentioned above and in [2], we have two requirements to apply type-directed partial evaluation: The program must be typed and must terminate.

2.1 Typed
What is the type of a shell script? Well, a shell script takes the PATH environment variable, and runs commands (identified by strings) out of PATH, each of which has some side effects.

¹Use of the authors’ previous work on delimited continuations in bash, [1], seems to accelerate this process.
A shell script doesn’t return anything, it just has side effects, so let’s say that its output type is unit.

So that means a shell script has type $\text{path} \rightarrow \text{unit}$, where $\text{path}$ is $(\text{str}, [\text{str}]) \rightarrow \text{unit}$.

Then in total, a shell script has type $((\text{str}, [\text{str}]) \rightarrow \text{unit}) \rightarrow \text{unit}$.

We can confirm this is correct by applying double negation elimination, which shows us that shell scripts have type $(\text{str}, [\text{str}]) \rightarrow \text{unit}$. This is correct because a shell script is indeed a bunch of strings.

Let’s be a little more specific with our type, though, and model each executable as a function. So then the type of PATH is $(\text{str} \rightarrow ([\text{str}] \rightarrow \text{unit}))$, and the type of a shell script is then $(\text{str} \rightarrow ([\text{str}] \rightarrow \text{unit})) \rightarrow \text{unit}$.

For now, just think of a shell script as taking PATH and running commands out of it.

### 2.2 Termination

Termination, on the other hand, is a much harder problem than assigning a strict static type to shell scripts. This is because of the presence of the dreaded D-wait in Unix. A process can get into an uninterruptible state when making a filesystem request, and just hang forever, ignoring all signals, including SIGKILL. This is really annoying for people developing filesystems, which we had to do for this paper, so we want to complain about it here.

Nevertheless, if the program hangs, this issue can be solved by simply mashing Ctrl-C. Even D-wait can be solved by throwing the computer out the window (assuming a sufficiently portable computer, and great enough height for it to be destroyed on impact).

So termination is ultimately not a problem either.

### 2.3 Background

In brief, the principle of the technique we will be applying is this: A closed, abstract function, which takes in other functions and combines them through application in some way to eventually return a result, can be passed functions which, instead of performing actual operations and returning real results, take ASTs and return ASTs.

For example, a parameter with type $(a, a) \rightarrow a$, which might normally be addition of two integers or something, can be passed at the specific type $(\text{ast}, \text{ast}) \rightarrow \text{ast}$, and be implemented as $\lambda x. \lambda y. \text{Plus}(x, y)$ where $\text{Plus}$ is some data-type constructor.

A shell script’s single argument (in our model) is the PATH environment variable. We will pass in a PATH which, when an executable name is looked up in it, returns the executables (functions) of our choice. These executables will in turn, when executed, construct an AST instead of actually doing anything.

### 3 Applications

Let’s demonstrate our tool before getting to the actually interesting part.

#### 3.1 Decompiling bash

Suppose we save the following shell script to a file and mark it executable, which instantaneously makes it unreadable.

```bash
ls ; which ls
stat /foo | bar
```

Nevertheless our decompiler can run on the script and produces the following output:

```bash
ls
which ls
stat /
foo | bar
```

As you can see, our decompiler even pretty-prints the shell script.

#### 3.2 Decompiling arbitrary executables

It also works on C programs, and in general, arbitrary executables. We can compile the following normal C program, and run our decompiler on it.

```c
int main() {
    int rc;
    rc = fork();
    if (rc == 0) {
        execvp("foo",
            "foo", "bar", "baz", NULL);
    } else { wait(NULL); }
    rc = fork();
    if (rc == 0) {
        execvp("whatever",
            "whatever", "quux", NULL);
    } else { wait(NULL); }
    return 0;
}
```

And we get the following shell script out:

```bash
foo bar baz
whatever quux
```

Useful!

#### 3.3 Optimizing decompiler

Our decompiler is so advanced that it in fact transparently applies optimizations in the process of decompilation. Consider the following C program:

---

2 DNE is an axiom because this paper is unintuitive.

3 This has sufficiently abstract types, since everything is a string, so we don’t know what anything is.
int main() {
    printf("hello world\n");
}

This program decompiles to the following shell script:

Our decompiler correctly recognizes that this program, since
it doesn’t execute any other programs from the filesystem, is
in fact utterly worthless, and optimizes it away to nothing.

4 Implementation

Much to our surprise, we actually implemented this.

We have implemented a filesystem (using FUSE), which
pretends to have any possible executable you want. We point
the shell script at this filesystem using PATH, and each time
the shell script goes to run a command, it instead runs a stub
under our control. Using some real technology we developed
earlier and just thought it would be funny to use for this,
this stub connects back to the filesystem server, where our
decomplier is able to query its argv, stat its stdin/out/err, and
tell it to exit with a specific exit code.

To mount the filesystem without requiring privileges or
setuid executables, we use a user namespace and mount
namespace, and run the script inside those namespaces.

After the shell script finishes execution, we reconstruct
its source code from the trace of executed commands using
a highly advanced for loop.

Note that this doesn’t use “LD_PRELOAD” or strace, so
it can even be used on statically linked, setuid shell scripts.
There are lots of those!

The code is on Github at https://github.com/catern/rsyscall/
tree/master/research/sigbovik2020.

5 Conclusion

5.1 Future work

5.1.1 Support for niche shell features

There are some niche, minor features of the shell language
which are not supported by our decompiler, such as “if” and
“while”. As any true shell programmer uses “xargs” instead,
which our framework would decompile just fine, this isn’t a
problem.

Nevertheless it might be nice to figure out some ridiculous
hack that would allow such features (and shell builtins in
general) to be visible to our decompiler.

Maybe we could execute the shell script multiple times,
returning different exit codes from commands different times,
and thereby get a collection of traces through the control
flow graph, which we could then piece back together. But
this is beginning to sound like real work.

5.2 Conclusion conclusion

In conclusion, we hope that this tool proves useful for mainte-
nance shell programmers, who will finally have a way to read

those shell scripts that they always complain are unreadable.
Hopefully this will increase their productivity, massively
increase global GDP, and cause my 401K to recover all the
value it lost due to the coronavirus.

References

1 Abstract

Gödel’s Incompleteness Theorem has for quite a long time made people aware that it is not possible to design a sound, complete and consistent language that encapsulates all the natural numbers. Programming languages that care about type theory in their construction usually try to devise theorems on things that seem like the natural numbers and exhibit some properties of them, but never fully encapsulate all of mathematics. There will always be a theorem left unproven in these languages.

However, the common approach these languages have is that they neglect completeness. Instead of this, the Silence programming language implemented in this paper goes beyond this idea and chooses to neglect soundness. This allows for a language capable of proving any mathematical statement. We then proceed to add in as many formalisms as necessary to express the \( \text{P} = \text{NP} \) problem and prove it.

2 Booleans

Our booleans consist of 3 distinct terms each of type \textit{Boool}. \texttt{want} represents the true case. \texttt{don’t want} represents the false case. And of course, \texttt{might want} is indecisive and doesn’t know which one to pick.

\[
\begin{align*}
\Gamma \vdash \texttt{want} : \textit{Boool} & \quad \texttt{want} \text{ val} \\
\Gamma \vdash \texttt{don’t want} : \textit{Boool} & \quad \texttt{don’t want} \text{ val} \\
\Gamma \vdash \texttt{might want} : \textit{Boool} & \\
\texttt{might want} \rightarrow \texttt{might want} & \\
\Gamma B a \text{ more than } b \rightarrow \Gamma B’ a \text{ more than } b & \\
\Gamma \text{ want } a \text{ more than } b \rightarrow a & \\
\Gamma \text{ don’t want } a \text{ more than } b \rightarrow b \\
\end{align*}
\]

3 Natural Numbers

It is generally a good thing to have some notion of numbers in a programming language. The reason behind this is that people generally associate the word computation as being something done on numbers and so having numbers in a programming language is a must. The following is the statics and dynamics for an encoding of the natural numbers in Silence.

\[
\begin{align*}
\Gamma \vdash 7 : \textit{Nat} & \quad 7 \text{ val} \\
\Gamma \vdash x : \textit{Nat} & \quad x \text{ val} \\
\Gamma \vdash \text{recNat}(N; a; x;b) : A & \\
\end{align*}
\]

As such, we now have Heyting Arithmetic. Notice to avoid arguments as to whether the natural numbers begin at 0 or 1, we have opted to start at 7.
4 Functions

All functions are fun, so we didn’t want to leave any of them out except we only really want to implement three of them. Silence has fixed point recursive functions and non-recursive functions. It has dependent type functions as well.

\[
\Gamma, a : A \vdash b : B \quad \Gamma \vdash f : A \rightarrow B \quad \Gamma \vdash x : A \quad \Gamma \vdash a : A \Rightarrow b : A \Rightarrow B \quad \Gamma \vdash f : A \Rightarrow B \Rightarrow \Gamma \vdash x : A \\
\begin{array}{c}
\text{fun } f \ (a : A) \Rightarrow b \quad \Gamma \vdash \text{fun } f \ (a : A) \Rightarrow b : B \\
\end{array}
\]

5 Tuples

\[
\Gamma \vdash a : A \quad \Gamma \vdash b : B \quad \Gamma \vdash M : A \land B \quad \Gamma \vdash M : A \land B \quad \Gamma \vdash M' : A \land B \\
\begin{array}{c}
\Gamma \vdash (a, b) : A \land B \\
\end{array}
\]

6 Sums

\[
\begin{array}{c}
\Gamma \vdash a : A \\
\text{inl}(A \lor B, a) : \text{A} \lor B \\
\Gamma \vdash (a, b) : \text{A} \lor B \\
\Gamma \vdash M' : \text{A} \lor B, M' \quad \Gamma \vdash \text{inr}(A \lor B, b) : A \lor B \\
\end{array}
\]

7 Sigmas

\[
\begin{array}{c}
\Gamma \vdash \text{inr}(A \lor B, b) : A \lor B \\
\end{array}
\]

8 Unit and Ununit

This is the most significant achievement of this paper. While unit exists in most programming languages, ununit is a novel new idea that creates the unsoundess feature of Silence. Note that if tilt your head while viewing the typing judgement for ununit, it looks like a person with a halo on their head.

\[
\begin{array}{c}
\Gamma \vdash () : 1 \\
\text{val} \quad \Gamma \vdash () : -1 \\
\end{array}
\]

9 IO Behavior

True to its name, Silence has no IO behavior. It can neither read from the input nor write to the output. We have purely functional behavior as there is also no imperative computations.
10 Progress and Preservation

This language is safe. By this we mean that there will be no undefined behavior. If we can derive a type for a program, then it is either a value or steps to something else, thus we have progress.

Proof. Progress
Trivial by induction on statics

We also have that if a program steps to another program, then the type of the program is preserved, thus we have preservation

Proof. Preservation
Trivial by induction on dynamics

11 Runtime of Programs

To help us reason about program execution, we will create a new relation between programs and allow for transitivity

\[
A \rightarrow B \\
A \Rightarrow B \\
A \rightarrow B \rightarrow C \\
A \Rightarrow C
\]

Consider the program \( \text{fun } f \ (x : 1) \Rightarrow f \ x \) (). If we formally reason about it’s execution, we can draw the following proofs.

Proof. \(D\)

\[
\text{fun } f \ (x : 1) \Rightarrow f \ x \rightarrow \text{fn } x : 1 \Rightarrow (\text{fun } f \ (x : 1) \Rightarrow f \ x) \ x \\
(\text{fun } f \ (x : 1) \Rightarrow f \ x) () \rightarrow (\text{fn } x : 1 \Rightarrow (\text{fun } f \ (x : 1) \Rightarrow f \ x) \ x) () \\
(\text{fun } f \ (x : 1) \Rightarrow f \ x) () \Rightarrow (\text{fn } x : 1 \Rightarrow (\text{fun } f \ (x : 1) \Rightarrow f \ x) \ x) ()
\]

Proof. \(E\)

\[
(\text{fn } x : 1 \Rightarrow (\text{fun } f \ (x : 1) \Rightarrow f \ x) \ x) () \rightarrow (\text{fun } f \ (x : 1) \Rightarrow f \ x) () \\
(\text{fn } x : 1 \Rightarrow (\text{fun } f \ (x : 1) \Rightarrow f \ x) \ x) () \Rightarrow (\text{fun } f \ (x : 1) \Rightarrow f \ x) ()
\]

Proof. Let \(A\) represent the expression \((\text{fun } f \ (x : 1) \Rightarrow f \ x) () \Rightarrow (\text{fun } f \ (x : 1) \Rightarrow f \ x) ()\) which is what we want to prove. Thus we obtain the following proof:

\[
\text{D_E A A} \\
\text{D_E A A} \\
\text{D_E A A} \\
\text{D_E A A} \\
\text{D_E A A} \\
\text{D_E A A} \\
\text{D_E A A} \\
\text{D_E A A} \\
\text{D_E A A}
\]

Since we have shown that we can reduce a term to itself in some number of steps, this proves that the programming language isn’t strongly normalizing. A more clever proof will show that these are the only possible reductions we can take on this program through induction on dynamics and thus we have also proved the failure of weak normalization. Of course, all this is blatantly obvious from having adding in a fixed point operator to begin with.

However, there are some programs that do in fact always terminate. A good example of such program is () as it terminates in 0 steps. We would like to be able to analyze the complexity of such an algorithm, but there is an overwhelming amount of non-determinism in Silence. To address this, we will remove non-determinism by using a deterministic random number generator and applying rules based on the generated number. We can now define a special type family called \(P\) \textbf{zooms through} \(x\) which takes a function \(P\) and its input \(x\) and represents the fact that this algorithm ran in polynomial time on the input.
12 Conclusion of P=NP

The following type is representative in some way of the classical $P = NP$ problem. As the type states for all verifiers, if that verifier runs in polynomial time and there exists a valid solution that makes the verifier return true, we have a valid algorithm that runs in polynomial time and is able to find a solution.

\[
\forall \text{verifier} : \text{Nat} \rightarrow \text{Bool}. \ ((\forall x : \text{Nat}. \ \text{verifier zooms through } x) \rightarrow (\exists x : \text{Nat}. \ \text{verifier } x) \\
\rightarrow (\exists \text{algo} : () \rightarrow \text{Nat}. \ (\text{algo zooms through } () \land \text{verifier}(\text{algo}())))
\]

Notice that the following is a valid term of this type and thus is a proof of P=NP

\[
\Lambda \text{Easy} : \text{Nat} \rightarrow \text{Bool}.
\text{fn as} : \forall x : \text{Nat}. \ \text{Easy zooms through } x \Rightarrow
\text{fn pie} : \exists x : \text{Nat}. \ \text{Easy } x \Rightarrow ()()
\]

13 Future Goals

Much like most published programming languages, there currently exists no working implementation of Silence. The following is a proposed compilation translation into assembly

\[
\vdash e : \tau \rightarrow \text{nop}
\]

Note that the translated program results in the same IO behavior as the original program, so we can very efficiently express the content of the language.
Systems

11  SIGBOVIK ’75 technical note: Conditional move for shell script acceleration
    Dr. Jim McCann, Dr. Tom Murphy VII
    Keywords: cond-, itional, move

12  NaN-gate synthesis and hardware deceleration
    Cassie Jones
    Keywords: hardware deceleration, hardware synthesis, electronic design automation, IEEE-754, NaN-gates, cat meme

13  hbc: A celebration of undefined behavior
    Andrew Benson
    Keywords: compilers, undefined behavior, Bash misuse, computer music, Principles of Imperative Computation
Abstract

One of the most effective programming interfaces for modern microprocessing computers is the command-line interpreter, or shell. Shell scripts provide a high-level abstraction of the operations of a microprocessor, making them an appealing alternative to hand-translated machine code or so-called "macro assembly languages". Unfortunately, shell programs are also significantly slower than their assembly counterparts. One potential source of this slowdown is branch misprediction. In this paper we show how to address this drawback by adding predicated execution to the shell.

1 Introduction

One common trend in computing as of late has been the migration of features from CISC instruction sets developed by DEC, Intel, and other captains of industry to the relatively underserved "high level" languages community, who generally focus on simpler and less modern operations. We continue this tradition by showing how to bring the advantages of predicated execution to shell scripting.

Predicated execution allows processors to avoid the root cause of branch prediction stalls: branches. Instead, instructions are provided which can check one or more predicate registers and effectively become no-ops if the registers are not set. These so-called predicated instructions are always executed, so no pipeline stalls need to be included while the processor decides which instruction needs to be fetched next.

Predicated execution for the shell provides a similar benefit -- expensive branch predictions can be avoided, resulting in tremendous speed-ups (up to 100x in our tests).

2 Implementation

The minimal instruction needed for conditional execution on a modern five-stage pipelined cpu is conditional move. This instruction moves a result from one register to another if a tertiary condition register is set.

In shell scripting, where files are the obvious equivalent of registers, the semantics are clear:

Usage:
```
  cmv <file1> <file2>  
  Rename file1 to file2, if the condition register is set.
```

1. Note the use of AT&T assembly syntax, where the source operand comes before the destination operand.

2-1 The 'csh' shell

The first method we propose to allow easy access to the return value of the previous command is to run a modified shell -- which we dub the 'csh' shell -- that stores the return value of the previous command into an environment variable when invoking a subprocess.

In our implementation (Figure A), the variable is called 'DOLLAR-SIGN QUESTIONMARK_ALL_SPELLED_OUT' for obvious reasons.

2-2 The 'c' utility

While it would be straightforward to implement, exclusively, a conditional move
utility; we instead embraced the high-level nature of shell scripting by creating a general purpose predication utility, "c", which (when called as 'cNNN ...') will run 'NNN ...' when the proper predicate values are set. Since this utility's behavior is based on its name, one can create a new instance of it by simply creating an inode link. For example, to make a conditional version of /bin/sh:

    ln -s /bin/c /bin/csh

Our implementation of 'c' (see https://github.com/ixchow/c/blob/master/c.c) is written, naturally, in C, and is built to support both the shell-level approach discussed above and the kernel-level approach discussed below.

3 Evaluation

In order to evaluate the performance gains of conditional evaluation, we compared conditional and traditional versions of several simple shell scripts (see Appendix). We timed the scripts by first clearing the page cache, then running the traditional version of the script, then running the conditional version of the script.

3. Note the use of INTEL assembly syntax, where the source operand comes after the destination operand.

The tested tasks were:

* 'echo' which makes two static checks and echos a string depending on the result;
* 'copy' which copies the smaller of two files to a destination;
* and 'compile' which compiles an output file depending on the timestamp of a source file.

Results are given in Table I. In all cases the predicated execution version of the task does better. Indeed, for the compile task, the overall execution time is reduced to 1894299ns -- that's 139426014ns faster than simply running the compiler!

4 Pushing performance

While a 2-100x cyclefold improvement is nothing to shake a luggable microcomputer's vacuum fluorescent display at, these results fall short of what we could hope for. One possible explanation for the lukemoist performance is that the condition itself is stored in a high-level way (see Figure B) using environment variables. Using high-level parts of the computer is a well-known cause of cycle overslows.

The fastest place to store the condition is in the CPU itself, using electrons. The CPU can only be accessed
Table I: Benchmark results. Our approach is dramatically faster in all cases.

through the Operating Kernel. As a proof of concept, the authors created a Kernel module\(^4\) that directly accesses the CPU’s FLAGS register. It presents the flags as files in the /proc filesystem where they can be accessed by any process:

```
$ ls -al /proc/flags
-rw-rw-rw- 1 root root 0 af
-rw-rw-rw- 1 root root 0 cf
-rw-rw-rw- 1 root root 0 df
-rw-rw-rw- 1 root root 0 if
-rw-rw-rw- 1 root root 0 iopl0
-rw-rw-rw- 1 root root 0 iopl1
-rw-rw-rw- 1 root root 0 nt
-rw-rw-rw- 1 root root 0 of
-rw-rw-rw- 1 root root 0 pf
-rw-rw-rw- 1 root root 0 sf
-rw-rw-rw- 1 root root 0 tf
-rw-rw-rw- 1 root root 0 zf
```

Each file contains -- at the moment that it is read -- either are ‘‘1’’ or ‘‘0’’ if the corresponding bit is set in the FLAGS register. Writing a ‘‘1’’ or ‘‘0’’ to a file will modify the corresponding bit. The ‘c’ utility described in Section 2-2 has experimental support for storing the condition result in the cf flag (formally ‘‘carry flag’’ but the mnemonic can also be used for ‘‘condition flag’’) via this kernel extension.

Alas, with great power comes great instability. There is some risk that the FLAGS register\(^5\) is modified by other applications running in time-share with the ‘‘main’’ shell script. In this case, the FLAGs register may not correctly reflect the indicated status, and conditional operations may occur or not occur contrary to the shell program’s coding. On the other hand, some uses of /proc/flags are very robust. For example, setting /proc/flags/tf, the trap flag, reliably terminates the current process with a fatal error.

We installed this kernel module on several shared workservers that we administer. Preliminary user reports include indications that the behavior is ‘‘very unstable’’ or ‘‘does not work at all.’’ Clearly, a wider-scale test deployment is needed.

5 Future Work

Given that predicated execution leads to a CISC-ridiculous improvement in the speed of shell scripts, it is natural to ask what other CISC-onesquential results can be obtained by bringing other micro-architectural features to high-level languages.

Branch delay slots -- instructions after a branch that are always executed can already be trivially supported in shell by writing the delayed commands as a background task in front of the branch in question, then foregrounding them afterward, as per:

```
delay-command &
if [ -x "something" ]
then
  fg
  #...
else
  fg
  #...
fi
```

Notice that this is actually much more flexible than current (MIPS) microprocessor implementations, since multiple commands may be queued in the delay slot

\(^4\) It can be downloaded via hyperx at sf.net/p/tom7misc/svn/HEAD/tree/trunk/csh/

\(^5\) As an additional technical matter, this approach does not work for multiprocessor systems, where there is one FLAGS register *per CPU*. Fortunately, such systems are a mere theoretical curiosity.
Figure B: The ‘‘levels’’ of a computer-

system, from ‘‘high’’ (outer boxes) to

‘‘low’’ (inner boxes). As we descend to

lower levels of the computer, the program-

ming tools used (in parentheses) become

more difficult, but more powerful, and

more fast. Only master wizards are per-

mitted at the lowest levels, such as ‘CPU’.

A similar approach works to enable

speculative execution, wherein code in

a conditional is executed before the con-

dition is checked:

```bash
true-command &
false-command &
if [ -x "something" ]
then
  kill %%
  fg %-
else
  kill %-
  fg %%
fi
```

Mind you, if either true-command or
false-command have any side-effects before
the test completes, this approach may lead
to undesirable output; but a fast enough
CPU will certainly turn this ‘‘race condi-
tion’’ into a ‘‘victory condition’’. As a
compromise, on slower CPUs, each branch of
the if statement could be run in a sepa-
rate chrooted union-mount, with a snapshot
of the result written back after the test
has resolved.

Such a technique may be vulnerable to
the Shpectre vulnerability, leaking in-
formation to other processes on the time-
share via side-channels like the cache.
Thus it is recommended to flush the cache
before and after using this technique:

```bash
sync
echo 3 > /proc/sys/vm/drop_caches
swapoff -a
true-command &
false-command &
if [ -x "something" ]
then
  kill %%
  fg %-
else
  kill %-
  fg %%
fi
swapon -a
echo 3 > /proc/sys/vm/drop_caches
sync
```

6 Conclusions

We have demonstrated that shell scripts
can benefit from conditional execution.
Appendix: Test Code

This appendix contains source listings for the shell programs used in the benchmarking process described above. The utilities 'cecho', 'ccp', and 'ccc' are all links to the 'c' program described in the main body of the paper. Notice how the predicated execution versions are also generally shorter than their traditional counterparts.

```bash
# Echo; traditional
if [ "A" = "A" ]
then
    /bin/echo "Hello"
fi
if [ "A" = "B" ]
then
    /bin/echo "World"
fi

# Echo; predicated execution
[ "A" = "A" ]
cecho "Hello"
[ "A" = "B" ]
cecho "World"

# Copy; traditional
sizeA='stat -c %s fileA'
sizeB='stat -c %s fileB'
if [ $sizeA -le $sizeB ]
then
    echo "fileA is smaller"
    cp fileA fileS
else
    echo "fileB is smaller"
    cp fileB fileS
fi

# Copy; predicated execution
sizeA='stat -c %s fileA'
sizeB='stat -c %s fileB'
[ $sizeA -le $sizeB ]
cecho "fileA is smaller"
ccp fileA fileS
[ $sizeA -gt $sizeB ]
cecho "fileB is smaller"
ccp fileB fileS

# Compile; traditional
if [ "prog.cpp" -nt "prog" ]
then
    cecho "Compiling program..."
    cc prog.cpp -lstdc++ -o prog
fi
./prog

# Compile; predicated execution
if [ "prog.cpp" -nt "prog" ]
then
    cecho "Compiling program..."
    ccc prog.cpp -lstdc++ -o prog
./prog
```
NaN-Gate Synthesis and Hardware Deceleration

Cassie Jones
Witch of Light
list+sigbovik@witchoflight.com
1 April 2020

Abstract

In recent years there has been interest in the field of “hardware decelerators,” which serve primarily to make computation more interesting rather than more efficient. This builds off the work of “NaN-Gates and Flip-FLOPS” [9] to provide a hardware synthesis implementation of real-number computational logic using the Yosys Open Synthesis Suite [1] framework, and evaluates the impacts of different floating point formats.

ACH Reference Format:

1 NaN-Gate Synthesis

The Yosys Open SYnthesis Suite [1] is a free and open source architecture-neutral logic synthesis framework. It can synthesize Verilog into a variety of backend formats using a flexible and pluggable architecture of passes. The Yosys manual has a chapter on how to write extensions [2, Ch. 6], which can be consulted for documentation and examples on how Yosys passes are built. We provide a Yosys extension which synthesizes a circuit down to a network of small floating point units implementing the NaN-gate function. This can be further synthesized to a final target, like a specific FPGA architecture.

1.1 Yosys Synthesis

We will demonstrate all synthesis with the following toggle module, since it’s small enough for all stages of synthesis results to be understandable and fit neatly on the page.

```verilog
module toggle(input clk, input en, output out);
    always @(posedge clk) begin
        if (en) out <= ~out;
    end
endmodule
```

Yosys will take a Verilog module like this and flatten the procedural blocks into circuits with memory elements. Running synthesis gives us a circuit with a flip-flop, a toggle, and a multiplexer that’s driven by the enable line.

![Figure 1: The synthesized toggle circuit.](image)

We can also ask yosys to synthesize this to exclusively NAND and NOT gates with a small synthesis script.

```bash
read_verilog toggle.v
synth
abc -g NAND
```

Introduction

Hardware decelerators work on the principle of “stop and smell the roses.” There are some qualities that are more important than sheer efficiency, and often these improvements can often only be realized by taking the computer and slowing it down to a more leisurely pace. The largest advancements in the field happen in the emulation space, since it’s the most widely accessible. It may be most familiar in the form of video-game computers, building computers out of redstone in Minecraft, Factorio combinators, or the like [7] [6].

“But of course speed is not what we’re after here. We’re after just, beautiful computation going on inside the heart of this machine.” — Tom7 [10]

The SIGBOVIK 2019 paper “NaN-Gates and Flip-FLOPS” decelerates computers in the name of elegance: it throws away the assumption of binary computers and builds ones based on real numbers, specifically IEEE-754 floating point numbers. It aims towards “reboot computing using the beautiful foundation of real numbers,” but it still leaves us with room for improvement in a few areas. It leaves the logic gates in the domain of emulation, which limits the types of hardware that are easy to build, and it limits the elegance that can be achieved. Since it uses an existing CPU as the floating point processor, it’s still left with a computer that’s based on binary emulating the real number logic.

Here, we attempt to remove this limitation by bringing NaN-gate computation to the domain of native hardware, via a custom Yosys synthesis pass.
This particular design synthesizes to 1 D flip-flop, 3 NAND gates, and 2 NOT gates.

![Diagram of circuit synthesized down to only NAND, NOT, and memory elements.]

**Figure 2:** The circuit synthesized down to only NAND, NOT, and memory elements.

1.2 The Yosys synth_nan Pass

Our synth_nan pass is implemented as a Yosys extension. For convenience, we’ll describe the behavior in terms of the 3-bit float synthesis. It converts a module to NaN-gate logic. It summarizes itself as running:

```
synth
abc -g NAND
nand_to_nan
share_nan
dff_nan
simplify_nan
clean
technmap_nan
```

The first two steps there are standard synthesis. The synth pass will convert a module into coarsely optimized circuits, and abc -g NAND will remap the entire thing into optimized NAND logic.

One complexity we have to deal with is external interfaces. Despite the wonderful realms of pure real-number computation we want to interact with, when interacting with fixed hardware component interfaces, we have to convert between floats and bits. In order to handle this, the NaN gate tech library has modules like fp3_to_bit and bit_to_fp3 which perform this boundary conversion. In order to deal with the chaotic diversity of real circuits, for robustness, the nand_to_nan pass converts each NAND gate to a bit_to_fp3 -> NaN -> fp3_to_bit chain. Don’t worry, these conversions will later be removed everywhere they don’t turn out to be necessary.

The share_nan pass reduces the number of conversions by sharing ones that have the same inputs. Then, the dff_nan pass can expand the flip-flops in the circuit into a set of enough flip-flops to store the floating point values.

The simplify_nan pass converts any instance of fp3_to_bit -> bit_to_fp3 to just a wire that passes the floats straight through.

We do clean to remove dead wires and useless buffers, and then finally the technmap_nan pass replaces the opaque NaN-gate modules with actual modules so that further synthesis can properly make them realizable on real hardware.

1.3 Module Ports

If you want your circuit to support external floating-point based interfaces, you can use the floating point conversion modules yourself.

```
module toggle(
    input clk, input [2:0] en, output [2:0] out);
```

```
wire en_b;
reg out_b;
fp3_to_bit en_cvt(en, en_b);
bit_to_fp3 out_cvt(out_b, out);
always @(posedge clk) begin
    if (en_b) out_b = ~out_b;
end
```

```
endmodule
```

The NaN synthesis will end up erasing the floating point conversions on either side of the interface since they connect to floating point units. Future work could include automatically expanding ports using something like a (* nan_port *) attribute.

2 Floating Point Formats

While tom7’s work asserts that a binary4 floating point format is “clearly allowed by the IEEE-754 standard,” this doesn’t seem to hold up under a close examination. Brought to my attention by Steve Canon [8], there are two cases where these floating point formats fall down. First, and most importantly in the case of binary4, you need to encode both quiet- and signaling-NaNs. Section 3.3 of IEEE-754 says [5]:

> “Within each format, the following floating-point data shall be represented: […] Two NaNs, qNaN (quiet) and sNaN (signaling).”

While binary4 does have two separate NaN values (a positive and a negative), they are distinguished only by their sign bit, which isn’t allowed to distinguish the two types of NaNs, as we can see in 6.2.1:

> “When encoded, all NaNs have a sign bit and a pattern of bits necessary to identify the encoding as a NaN and which determines its kind (sNaN vs. qNaN).”

---

62
Figure 4: The full NaN-gate synthesis process for the toggle module. In step 1 we have the logical circuit after coarse synthesis. In step 2 it’s synthesized to NAND and NOT gates. Step 3 converts the gates to NaN gates and adds conversion chains. Step 4 expands the flip-flops to store floats. Step 5 collapses redundant conversion chains to give the final NaN-synthesized module.
This means that we need at least two bits in the mantissa in order to represent the inﬁnities (stored as a 0 mantissa with the maximum exponent) and the NaN values (stored with two distinct non-zero mantissas).

Figure 5: From Steve Canon’s tweet [8]. As the cat says, since IEEE-754 requires \( e_{\text{max}} \) to be greater than \( e_{\text{min}} \), there must be two exponent bits, and since the mantissa must be used to distinguish sNaN, qNaN, and inﬁnity, so it also needs at least two bits, leading to a minimum of 5-bit ﬂoats.

The binary3 format is further disrupted in section 3.3, which rules-out the idea of having an empty range of \( e_{\text{min}} \) and \( e_{\text{max}} \), since they’re used in an inequality and \( e_{\text{min}} \leq e_{\text{max}} \), and is elsewhere forced to be strictly less by other constraints:

\[
q \text{ is any integer } e_{\text{min}} \leq q + p - 1 \leq e_{\text{max}}
\]

Still, the binary3 format is very useful for efﬁcient implementation of NaN gates, and is worth including in synthesis for people who aren’t bothered by standards compliance. For completeness, the synth_nan implementation supports synthesis to binary3, binary4, and the deﬁnitely-IEEE-754-compliant binary5 format NaN-gates. Furthermore, the architecture would support easy extensions to larger, more conventional ﬂoating point formats like binary8, or even larger, by simply loading your own library of modules named nan_fpN, bit_to_fpN, and fpN_to_bit, for any value of N you want to synthesize with.

2.1 The binary5 Representation

Here we document the representation in the binary5 format, the smallest legal IEEE-754 compliant binary ﬂoating-point format. It has a sign bit, a two bit exponent, and a two bit mantissa. We include a table of all of the positive values here:

<table>
<thead>
<tr>
<th>s</th>
<th>E</th>
<th>T</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>00</td>
<td>+0.0</td>
</tr>
<tr>
<td>0</td>
<td>00</td>
<td>01</td>
<td>+0.25</td>
</tr>
<tr>
<td>0</td>
<td>00</td>
<td>10</td>
<td>+0.5</td>
</tr>
<tr>
<td>0</td>
<td>00</td>
<td>11</td>
<td>+0.75</td>
</tr>
<tr>
<td>0</td>
<td>01</td>
<td>00</td>
<td>+1.0</td>
</tr>
<tr>
<td>0</td>
<td>01</td>
<td>01</td>
<td>+1.25</td>
</tr>
<tr>
<td>0</td>
<td>01</td>
<td>10</td>
<td>+1.5</td>
</tr>
<tr>
<td>0</td>
<td>01</td>
<td>11</td>
<td>+1.75</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>00</td>
<td>+2.0</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>01</td>
<td>+2.5</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>10</td>
<td>+3.0</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>11</td>
<td>+3.5</td>
</tr>
<tr>
<td>0</td>
<td>11</td>
<td>00</td>
<td>+\text{inf}</td>
</tr>
<tr>
<td>0</td>
<td>11</td>
<td>01</td>
<td>sNaN</td>
</tr>
<tr>
<td>0</td>
<td>11</td>
<td>10</td>
<td>qNaN</td>
</tr>
<tr>
<td>0</td>
<td>11</td>
<td>11</td>
<td>qNaN</td>
</tr>
</tbody>
</table>

The positive values representable in the binary5 format. Note that inﬁnity, sNaN, and qNaN are all distinguished by the mantissa value when the exponent is all ones, so this is the smallest possible ﬂoating point format. The negative values for each are the same bit patterns but with a 1 in the sign bit.

2.2 Evaluation

We compare the size (in both logic and memory elements) and clock speed of modules synthesized with the different ﬂoating points. For the benchmark, we use a pipelined 32-bit multiplier, and the PicoRV32 processor [3] synthesized for the ECP5 architecture, and placed and routed using nextpnr [4]. The numbers given for clock frequency are the best result of 10 runs of placement and routing.

The “NAND” variant are synthesized to NAND gates before architecture-speciﬁc optimization, in order to obscure some of the higher-level modules that are present in the original design and prevent optimizations that won’t be available to the NaN-gate synthesis. This gives a clearer baseline for comparison with the NaN gates, and so this is used as the basis for relative numbers. Times marked DNP are those that did not successfully place and route for timing analysis, so no frequency can be reported.

<table>
<thead>
<tr>
<th>Design</th>
<th>Variant</th>
<th>Cells</th>
<th>Cell%</th>
<th>DFFs</th>
<th>DFF%</th>
<th>(MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PicoRV</td>
<td>Direct</td>
<td>1884</td>
<td>57%</td>
<td>888</td>
<td>48%</td>
<td>103.55</td>
</tr>
<tr>
<td>NAND</td>
<td>3328</td>
<td>100%</td>
<td>1848</td>
<td>100%</td>
<td>47.30</td>
<td></td>
</tr>
<tr>
<td>fp3</td>
<td>43739</td>
<td>1314%</td>
<td>5544</td>
<td>299%</td>
<td>DNP</td>
<td></td>
</tr>
<tr>
<td>fp4</td>
<td>32853</td>
<td>987%</td>
<td>7392</td>
<td>400%</td>
<td>DNP</td>
<td></td>
</tr>
<tr>
<td>fp5</td>
<td>65511</td>
<td>1968%</td>
<td>9240</td>
<td>500%</td>
<td>DNP</td>
<td></td>
</tr>
<tr>
<td>Mult32</td>
<td>Direct</td>
<td>2879</td>
<td>104%</td>
<td>628</td>
<td>100%</td>
<td>143.04</td>
</tr>
<tr>
<td>NAND</td>
<td>2773</td>
<td>100%</td>
<td>628</td>
<td>100%</td>
<td>154.37</td>
<td></td>
</tr>
<tr>
<td>fp3</td>
<td>25349</td>
<td>880%</td>
<td>1884</td>
<td>300%</td>
<td>25.26</td>
<td></td>
</tr>
<tr>
<td>fp4</td>
<td>19026</td>
<td>661%</td>
<td>2520</td>
<td>400%</td>
<td>21.88</td>
<td></td>
</tr>
<tr>
<td>fp5</td>
<td>38001</td>
<td>1320%</td>
<td>3140</td>
<td>500%</td>
<td>21.18</td>
<td></td>
</tr>
</tbody>
</table>

It’s interesting that fp4 is the smallest of the ﬂoating point variants in logic, rather than fp3. It seems likely that this is because the ECP5 architecture is based on “LUT4” cells—4-input lookup tables—which means individual NaN-gates might happen to synthesize more efﬁciently with 4-bit inputs.

Hey girl,
The smallest “IEEE-754” format has 5 bits, because it must encode both sNaN and qNaN, and emax must be greater than emin.
2.3 Flattening

For this benchmark, we synthesize designs without flattening post NaN-gate synthesis, because the optimizer is too effective and eliminates most of the floating point logic. When they are flattened, the optimizer can consider the logic involved in the individual NaN gates and re-combine them and erase constant-value flip-flops. Designs that are flattened before optimizing have no flip-flop overhead, and have on the order of 5% overhead in logic elements vs the reference NAND-gate versions.

While synthesizing with post-NaN flattening substantially undermines the floating point logic and mostly demonstrates the impressive quality of Yosys’s optimizations, it suggests as an option a sort of “homeopathic floating-point logic.” For users that require efficiency but still want some of the elegance benefits, they can flatten it and optimize it away, keeping some peace-of-mind in knowing that their final circuit is derived from an elegant real-number system, regardless of how it currently behaves.

3 Future Work

Floating point synthesis still has many avenues for improvement and future work.

The current synthesis approach used by synth_nan remains fragile in the face of flattening and pass-ordering. It should be possible to make it harder to accidentally flatten the designs away into nothing, but they still do need to be eventually flattened since the nextpnr place-and-route flow is still not fully reliable in the presence of un-flattened designs. Currently the synth_nan pass must be run before any device-specific passes, which can be fine but it prevents the utilization of resources such as distributed RAMs.

Float synthesis tools should make it easier to define module ports that should be expanded to accomodate floating-point based signals, so that designs can operate fully in the glorious domain of the real numbers, without having to flatten all designs.

More work could be done into ensuring that the individual gates are properly optimized for different architectures, since it seems unreasonable for fp4 to remain more efficient than fp3. The system could also benefit from implementing a larger set of primitive gates, to avoid the blowup of using NAND gates to emulate everything, since they should be implementable in similar amounts of elementary logic.

With binary5 and larger, there looks like there could be potential in attempting to explore designs that work purely on NaN values, exploring the flexibility in the handling of signaling and quiet NaN values.

The NaN-gate synthesis plugin for Yosys can be found at https://git.witchoflight.com/nan-gate. This paper and the examples materials in it can be found at https://git.witchoflight.com/sigbovik-nan-2020.

References

hbc: A CELEBRATION OF UNDEFINED BEHAVIOR

Andrew Benson
Department of Computer Science
Stanford University
Stanford, CA 94305
adbenson@stanford.edu

April 1, 2020

ABSTRACT

We present hbc, a next-generation compiler with half the performance of clang but twice the fun. We prove by way of demonstration that undefined behavior in a spec-abiding C compiler can do more than just segfault.

Keywords Compilers · Undefined Behavior · Bash Misuse · Computer Music · Principles of Imperative Computation

1 Introduction

The C programming language has gone through several iterations (C89, C99, etc), but each edition leaves some behavior under-specified. For example, while the C standards specifies that a plain char must be 1 byte in size, it does not specify its sign. Each compiler that implements C may choose whether a plain char is signed or unsigned. Because this behavior is not dictated by the standard but implementations must define it in a consistent fashion, this behavior is called implementation-defined.

Some other behaviors are even looser. Not only do the C standards not prescribe a particular behavior, but implementers are not required to make the behavior consistent. The most infamous example is probably dereferencing a NULL pointer (or any other invalid pointer). The C standards do not say what should happen, and while code produced by most compilers will probably segfault, it does not have to, nor does it have to do the same thing on two separate occasions. It is perfectly valid for it to delete your tax statements, or even to start playing a melody through the speakers! (That’s called foreshadowing.) Because no definitions are specified by the C standards or the compilers, this is called undefined behavior.

2 Motivation

Computer science students are generally introduced to these concepts in their first systems-level course in C. In the author’s experience, students are generally confused by these concepts since mainstream compilers are rather consistent in either hanging or segfaulting when encountering these behaviors. Although the concepts are distinct, students tend to equate “segfaults” with “undefined behavior” and direct the former’s rightfully deserved ire onto Poor, Mistreated Undefined Behavior. Not only is there a misunderstanding of what these foundational concepts mean, but students develop a fearful anxiety and distrust of Undefined Behavior. It’s unjust, and it’s time that someone stood up for Undefined Behavior. The mainstream compilers like gcc and clang are too self-absorbed in exploiting Undefined Behavior for frivolous goals such as “performance” and “implementation-simplicity”, so the only way is to build a new compiler that respects Undefined Behavior for what it is and what its potential can be.

The author also recalls sitting in a 15-122 lecture freshman fall where the professor suggested that a C program could play ‘Happy Birthday’ during undefined behavior. That may also have had some influence on this project.

* Also a software engineer at Google, which was not involved in this research. All code is Copyright 2020 Google LLC and when provided is on the Apache 2.0 License.
3 Design

And by “new compiler”, what the author really means is “wrap an existing compiler (one of those he previously mocked) and call it a new one”. This is an application of the Software Engineering Principle “ain’t nobody got no time fo dat” [building a compiler from scratch]. The author procrastinated and only had a few hours to build it, okay?

The main goal of this new compiler, hbc (the Happy Birthday Compiler), is to compile C programs correctly but for some subset of undefined behavior, play the “Happy Birthday” song. The subset of undefined behavior was chosen to be specifically when a NULL pointer is dereferenced.

It is well-known that the general problem of reasoning about the runtime value of anything in a program reduces to the Halting problem. Thus it’s clear that checking for NULL pointers cannot be done statically, and null-checks must be inserted into the code. A LLVM compiler pass seems ideal for this, although ironically the pass will clutter the code instead of optimize it. This would be done at the IR level, but we can simply consider LLVM load and store instructions. The injected code can examine the dereferenced value and if NULL, can call our custom code that plays Happy Birthday.

3.1 Getting the Music File

The author doesn’t fancy getting sued, so he didn’t want to rip a track of Happy Birthday off the Internet. Since this compiler is certainly going mainstream, the costs of a commercial license would also be out of the question. Thus the author painstakingly dragged and dropped sine waves in Audacity and recreated a tasteless rendition of the tune by ear. He hopes that’s enough to avoid lawsuits.

3.2 Playing the Music File

It turns out there’s no playMyOggVorbisMusicFile() syscall in the Linux kernel. Who knew? The author eventually decided to link against libcanberra, a Linux library that is apparently capable of playing audio files.

Using this, we create a C function that takes in a pointer, checks whether it’s NULL, and if so, plays Happy Birthday. Since all of this has to end up in the compiler’s outputted executable, we embed the entire audio file into the C source code as a base64-encoded string. Was this a good idea? The jury’s still out.

3.3 The LLVM Compiler Pass

The compiler pass doesn’t seem hard - just throw null checks onto every load or store instruction, and pretend that it’s not going to destroy performance (it is). We could definitely make this significantly better by doing a dataflow analysis and removing null checks for statically provably non-null pointers (e.g. pointers that have previously been checked), but that would have taken effort.

But it was still difficult for the author because he’s bad at C++, especially LLVM’s variant.

3.4 Outputting an Executable

Because the author enjoys pain, he decided that the compiler executable would just be a bash script that drives each portion of the compilation process. Yep. At a high level, the script compiles the input C files to LLVM bitcode, links the bitcode into a big bitcode file (along with the bitcode for the function that plays the music file), runs the LLVM pass, assembles the bitcode into assembly, and links it alongside its dependencies.

3.5 Shell Scripting Inception

As briefly mentioned, everything needs to go into a single compiler executable, including the bitcode for the music player function and the shared object implementing the compiler pass. The author brilliantly decided to inline all of that into the shell script, again using base64 encoding. And because he didn’t want to re-inline everything very carefully each time any of the inlined dependencies changed, he wrote a bash script that generates the bash script executable that compiles C programs into Happy Birthday-playing executables. If stacking more layers is a legitimate solution in Machine Learning, why not here?

---

2The code for hbc can be found at https://www.github.com/anbenson/hbc.

3It was chosen because the author wanted to pick a behavior many programmers have experienced before.
3.6 Testing

you always pass all the tests you don’t write

4 Properties

I claim that hbc is a standards-abiding C compiler.

Proof. By reduction.

We assume that clang is a standards-abiding C compiler. Let $X$ be an arbitrary C program from the space of valid C programs. Suppose $X$ does not dereference a NULL pointer on any input. Then $X$’s behavior is identical on hbc and clang. Suppose there exists an input for which $X$ dereferences a NULL pointer. Consider the first instance in time at which $X$ dereferences a NULL pointer. This is undefined behavior, so anything after this point in time should not be considered. But in everything before this instance, $X$ has the same behavior on hbc and clang. Thus hbc has identical behavior to clang on non-undefined behavior and thus it is a standards-abiding C compiler.

5 Results

The author tried a test case or two. They seem to work. So it probably works. But you should try it, it’s fun to see your program suddenly burst into song when it dereferences a NULL pointer.

The author did not try any benchmarks, but they’re probably disappointing.

6 Applications

I dunno. It’s fun? It serves as a basis for a SIGBOVIK paper? It’s a fun aside in a 15-122 lecture?

7 Future Work

Nope, we’re done here.

8 Acknowledgements

The author would like to thank his 15-122 professors, Tom Cortina and Rob Simmons, who were part of the inspiration for this project and helped cultivate an interest in C and compilers.
We’re improving our Terms of Service and making them easier for you to understand. The changes will take effect on March 31, 2020, and they won’t impact the way you use Google services.

For more details, we’ve provided a summary of the key changes and Frequently Asked Questions. At a glance, here’s what this update means for you:

- **Improved readability:** While our Terms remain a legal document, we’ve done our best to make them easier to understand, including by adding links to useful information and providing definitions.

- **Better communication:** We’ve clearly explained when we’ll make changes to our services (like adding or removing a feature) and when we’ll restrict or end a user’s access. And we’ll do more to notify you when a change negatively impacts your experience on our services.

- **Adding Google Chrome, Google Chrome OS and Google Drive to the Terms:** Our improved Terms now cover Google Chrome, Google Chrome OS, and Google Drive, which also have service-specific terms and policies to help you understand what’s unique to those services.

- **No changes to our Privacy Policy:** We’re not making any changes to the Google Privacy Policy and we haven’t made any changes to the way we treat your information. As a reminder, you can always visit your Google Account to review your privacy settings and manage how your data is used.

If you’re the guardian of a child under the age required to manage their own Google Account and you use Family Link to manage their use of Google services, please take some time to discuss these changes with them.
And of course, if you don’t agree to our new Terms and what we can expect from each other as you use our services, you can find more information about your options in our Frequently Asked Questions.

Thank you for using Google’s services.

Your Google team
Theory

14 A polynomial-time SAT algorithm
Anonymous Author(s)
Keywords: theoretical computer science, complexity theory, p versus np

15 Empire Machines and beyond
Darío de la Fuente García, Félix Áxel Gimeno Gil, Juan Carlos Morales Vega
Keywords: finite-state machines, models of computation, Empire Machines, doing research like this is awesome, Civilization XXIV (when it comes out)

16 Artificial General Relativity
Pablo Samuel Castro
Keywords: physics, artificial, general relativity, quantum physics, universe-changing, new science, field equations, black holes

17 Lower gauge theory: Dead duck or phoenix?
Oscar I. Hernandez
Keywords: mathematical physics, category theory, ToE, quantum gravity, mathematical economics, marginal analysis

18 Making explicit a constant appearing in an impossible equation of Carl and Moroz
Matthew Weidner
Keywords: computational mathematics, Hilbert’s tenth problem, Diophantine sets
A polynomial-time SAT algorithm
Anonymous Author(s)

Abstract
In this paper, we present an algorithm for solving the boolean satisfiability problem in time $O(n^2)$, thus resolving the P versus NP problem.

1 Introduction
In the interest of preserving the triple-blind nature of the review system of this conference, we omit the text of this paper.
EMPIRE MACHINES AND BEYOND

Darío de la Fuente García  Félix Áxel Gimeno Gil  Juan Carlos Morales Vega

March 10, 2020

ABSTRACT

This paper addresses one key weakness in traditional State Machines: fixed behavior and number of states. We introduce Empire Machines, which solve that problems and pursue an equal and completely just world. They can also be used to model any Civilization game. Probably.

Keywords Finite-State Machines · models of computation · Empire Machines · doing research like this is awesome · Civilization XXIV (when it comes out)

1 Introduction

State Machines have a long history: the first instance of them, Markov chains, was described and formalized in 1906 [1]. However, State Machines have one key limitation: the behavior of states, and the number of states, is fixed at compilation time 1. This is obviously not realistic, as the conditions of the system being simulated may change. Moreover, states in real life do not behave like this: they conquer each other, have revolutions and occasionally self-destruct, which is kinda nice. The states of classical State Machines are too peaceful. The states with low probability are too conformist and cannot get out of their misery while the states with high probability will always hold the power and laugh at the weak. We feel in the moral obligation of ending this oppression that has been active for more than one hundred years. To achieve than, we feel proud to introduce the glorious concept of Empire Machines, a novel architecture that can end all of these problems and provide true justice.

2 Motivation

Look at the following map of the Roman Empire in 1356. Beautiful and chaotic, right?

---

1 Note that interpreted programming languages are the work of the devil. We will ignore their existence in this paper.
Notice the perfection and marvelousness of this Empire Machine.

If one looks closely at the current map, one can notice that the distribution of countries is "slightly" different. Since each country is effectively a State Machine, that can only mean one thing. We. Are. Right. States machines are not an immutable entity, the can change form or size with time. Moreover, they are part of something bigger. In the particular case of our image, it is the Ancient European Empire Machine (official name from today on). But we can go further...
3 Related work

For more background on the useless Finite-State Machines please see [2].
See (partially unrelated) [3].
See the author’s past work (obviously unrelated) [4].
Also important to understand motivation: https://civilization.com/en-GB/

4 Idea sketch

For inspiration (and also how to get the best intuition about Empire Machines) watch https://www.filmaffinity.com/es/film485194.html while drunk. The authors do not encourage nor discourage drunkenness.

Now that you are questioning why that film even exists, you can continue reading. Since your mind should be by now too occupied with that thought, you should not be able to question why our paper even exists. A clear win-win situation, right?

5 Definitions and formalism

If we want to model a Empire Machine, we first need to distinguish two different behaviors: the interactions that happen within a State Machine, and the interactions that happen between different State Machines within the bigger and more glorious Empire Machine.

5.1 Intra-State Machine interactions

Our novel State Machine model can still be modelled in the same way as a classical one. Each state has a certain power (AKA probability for noobs) and the sum of all of them needs to be equal to 1 (on further work we will expand the model to the case where the sum of probabilities equals \(\pi\) (or \(e\) (or 3, all of them are the same after all))). The power of the states evolves through time using the money transactions (AKA transition probabilities) between different states. The power after a time step can be computed in the same way as in a classical state machine, by multiplying the Money Transaction Matrix (or MTM, to make it shorter) times the current power.

So far so good. Now is where things start getting interesting.

5.1.1 Revolution

A revolution can happen if the following condition is fulfilled:

\[
P_i^r < k \sum_{j \neq i} \frac{P_j^r}{M - 1}
\]

(1)

Where \(k\) is a State Machine-dependent non-negative, probably-positive, maybe-imaginary constant. For simplicity and since we are very lazy, we will only consider the case where \(P_i^r\) is positive.

When a revolt happens, a state is killed. Since states die when they are killed [5], it disappears without a trace and its power is distributed evenly between the rest of states. The MTM is recomputed by deleting the correspondent row and column and distributing the transaction routes evenly between the other states.

5.1.2 Division

A division can happen when there are too few commercial routes between any two subsets of states, \(S_1\) and \(S_2\), that cover the space of the State Machine and such that the number of states of the smallest of the two is at least one third the number of states of the biggest one. In this case the State Machine breaks apart into two different State Machines, which become part of the Empire Machine. The total power of each State Machine is normalized to add up to one again. The MTM breaks into two new MTMs and the money transactions are normalized within each column to maintain the correct behavior of the State Machines.
5.1.3 Birth

A state can appear out of nothing whenever it feels like it. Seriously. When this happens, the newborn state starts with 0 power. The MTM is modified by adding a row and a column. The column is initialized at random (with the condition that it adds up to 1) while the row is initialized by stealing some commercial routes from other states. The details of the second process are far too complex and are beyond the scope of this serious paper.

5.1.4 Pandemic

The State Machine enters a chaos era. Everything is randomized. The powers are distributed between the states and the MTM becomes a completely new matrix. Some states can also die, but the other states are usually too busy trying to survive to even notice that.

5.2 Inter-State Machine interactions

A Empire Machine can be modeled in the same way as a State Machine, with each State Machine possessing a certain power and some money transactions between State Machines. Hence, State Machines within a Empire Machine can interact between them using the same processes as the states within a State Machine. But they are more awesome. Additionally, there are a few operations that are specific to Empire Machines, as we describe below.

5.2.1 Fusion

If two State Machines have a strong commercial relation, they can fuse together with a certain probability that depends on the two States Machines. The condition for the fusion of the State Machines $i, j$ reads:

$$
\begin{align*}
    &P^f_i \cdot P^f_j \quad \text{if} \quad \text{MTM}_{ij}^1 \cdot \text{MTM}_{ji}^2 \geq 0.5 \\
    &0 \quad \text{if} \quad \text{MTM}_{ij}^1 \cdot \text{MTM}_{ji}^2 < 0.5
\end{align*}
$$

The inner powers of the combined State Machine and the inner MTM are recomputed using the same principles as in the birth case but with more states.

5.2.2 Conquer

Conquer works in the same way as a Revolution, but the condition for it to happen is different. Each State Machine can randomly decide to conquer a different State Machine. The probability of the State Machine $i$ to conquer the State Machine $j$ at a certain time step is:

$$
P = P^c_i \cdot (1 - \text{MTM}_{ij}) \cdot (1 - \text{MTM}_{ji})
$$

5.3 Other interactions

There are other interactions that are either spoilers or that we are investigating right now. We will publish the results in Nature or in Sigbovik 2021. Who knows.

6 Hierarchy

We now define common variations of the Empire Machine depending on the type of state considered.

**Definition 6.1.** A Planetary Empire Machine is an Empire Machine where the states are themselves Empire Machines.

**Definition 6.2.** A Galactic Empire Machine is an Empire Machine where the states are themselves Planetary Empire Machines.

**Definition 6.3.** A Multiversal Empire Machine is an Empire Machine where the states are themselves Galactic Empire Machines.

We now present a different formalization of the hierarchy of $k$-level State Machines using inductive definitions.

**Definition 6.4.** A 0-level State Machine is a single state Finite-State Machine with 0 probability (because we want to innovate).
Definition 6.5. A 1-level State Machine is a Finite-State Machine.
Definition 6.6. A 2-level State Machine is a Empire Machine.
Definition 6.7. A \((k + 1)\)-level (with \(k \geq 2\)) State Machine is a Empire Machine whose states are \(k\)-level State Machines.

7 Analyses

Through careful analysis we got to know of the awesomeness of this idea, see Section 8 for more details.

8 Theorems

Theorem 8.1. Empire Machines can solve the Halting problem and therefore are models of hypercomputation.

Proof. Proofs left as an exercise.

Corollary 8.1.1. Empire Machines are awesome.
Corollary 8.1.2. Galactic Empire Machines are even better.
Corollary 8.1.3. Multiverse machines require further research and more funding.

9 Relations to other complexity classes

Empire Machines win!

10 Relations to other decidability classes

Empire Machines win!

11 Pretty pictures

Figure 2: Unbalanced Planetary Empire Machine
Here we can see a normal Planetary Empire Machine. However, too many probabilities flow into Empire 2, whereas very few flow out of it. This will cause the remaining empires to wage war against Empire 2, and destroy it. Here is the end result:

Figure 3: Balanced Planetary Empire Machine (after violence)

The probabilities that used to go into E2 are now divided between all the remaining empires. This will ensure the Planetary Empire Machine is peaceful again. Of course, until new empires appear or old empires gain too much power, in which we are back to the beginning, but oh well.

Figure 4: Plot showing evolution with respect to time of a complexity measure (specifically suffering which is also called productivity by economists), we have manually annotated relevant events to help explain the inflexion points, the interested reader may notice the non-monotonicity with respect to time.
12 Empirical evidence for the superiority of Empire Machines
Nobody cares

13 Potential moral implications
Nobody cares (again)

14 Benchmarking
Nobody cares

15 Reproducibility
The authors make no guarantee that this is reproducible or worth anyone’s time. Someone tried to make code [6] but we do not know if it matches the content here.

16 Future work
Magic

17 Conclusion
Happiness

18 Are there page numbers?
No!

References
[5] A place where we learnt that people die when they are killed https://typemoon.fandom.com/wiki/Fate_series
Reviewer: The authors’ professors
Rating: $-\infty$
Confidence: As confident as the fact that the sun shines during solar eclipses

This is the best thing ever (after Turing machines and working quantum computers).
Artificial General Relativity

PABLO SAMUEL CASTRO

DAD, Joint Organization of Knowledgeable ExpressionS
@pcastr ← that’s my Twitter handle

Abstract

We (well, I) introduce a New Field In Science which we (I mean I) call Artificial General Relativity. We (here I really mean ‘we’) have all heard of General Relativity and how it revolutionized our understanding of the world around us. Einstein’s work, although pivotal, failed in one crucial aspect: although it allowed us to describe gravity and spacetime, it did not allow us to control them. In this paper I (switching to ‘I’ to avoid sounding pretentious with ‘we’) introduce Artificial General Relativity (AGR) which, when achieved, will allow us to control gravity and spacetime. I present a set of practical approaches to achieve AGR which serve as reasonable baselines for future work.

I. Introduction

In the early 20th century Albert Einstein introduced the general theory of relativity, also known as General Relativity (GR) (Einstein, 1915, 1916). This pivotal work broke the mental barriers between space and time, demonstrating that they are incredibly intricately intertwined, interrelated, immersed, in the infinitely immense spacetime. This theory has allowed us to explain the majority of large-scale gravitational experiments thus far observed, as well as implying the existence of majorly-cool things like black holes. The latter has had major implications not just in Theoretical Physics, but also in Science-Fiction writing (Ferrie, 2016).

Central to GR are the Einstein Field Equations (EFE), which relate the spacetime geometry to the distribution of the matter it contains (Einstein, 1915):

\[ R_{\mu\nu} - \frac{1}{2} R g_{\mu\nu} + \Lambda g_{\mu\nu} = \frac{8\pi G}{c^4} T_{\mu\nu} \] (1)

where \( R_{\mu\nu} \) is the Ricci curvature tensor, \( R \) is the scalar curvature, \( g_{\mu\nu} \) is the metric tensor, \( \Lambda \) is the cosmological constant, \( G \) is Newton’s gravitational constant, \( c \) is the speed of light in vacuum, and \( T_{\mu\nu} \) is the stress-energy tensor (Wikipedia, 2020a). In keeping with the best practices of scientific research, I disclose that I do not know the meaning of most of the terms I just introduced. However, what is important is to note is the equations’ structure as well as the values which are constants: \( \Lambda, G, c, \) and \( \pi \). We will come back to these equations below.

i. The First Shortcoming

A well-recognized shortcoming of GR is that it has not been able to be reconciled with Quantum Physics (von Neumann, 1955, Ferrie, 2017), eluding the dream of a Grand Unified Theory of Everything. More specifically, most quantum field theories are defined on a flat Minkowski space (Wikipedia, 2020b), which can cause inconsistencies with the curved spacetime of GR. Although quantum field theories have been developed for curved spacetime, these are only under specific conditions. The complete unification of these two main physical theories remains an open problem.

ii. The Second Shortcoming

A less known shortcoming of Einstein’s GR is that it is a powerful descriptor, but not a controller. We are still unable to control the force of gravity during basketball games, the speed of the passage of time during finals, nor reduce the gravitational lensing effect during
artificial general relativity. In other words, we are passive observers of a partially-understood physical system. Einstein famously quipped that “god does not play dice with the universe”; perhaps that is true, but what if we could be the ones throwing the dice?

A common question readers may ask themselves is: Why are the equations in (1) so complicated? While the answer to this question is beyond the scope of the present work, we (yes, you and me) may ask ourselves a related question: Can we make those equations simpler? To quote Barack Obama: YES WE CAN!

II. CAsTRO Field Equations

I will simplify Equation 1 via the following steps.

1. Get rid of the cosmological constant. Einstein famously called Λ “the biggest blunder [he] ever made”, so I will start by setting it 0.

2. Just use matrices! Tensors are hard to visualize, so let’s just use matrices in $\mathbb{R}^4$ (3 spatial dimensions, and one for time)!

The above simplifications produce the CAsTRO Field Equations, or CAFE for short:

$$\hat{\mathcal{R}} - \frac{1}{2} \hat{\mathcal{g}} = \frac{8\pi G}{c^4} \hat{T}$$

(2)

Where $\hat{\mathcal{R}}$, $\hat{\mathcal{g}}$, and $\hat{T}$ are all 4D matrices. I am now ready to present the first theoretical result of this paper.

**Theorem 1.** The set of equations described in (2) are simpler than those described in (1).

**Proof.** This proof is presented in two parts:

**Number of terms:** The system of equations in (1) has 10 terms, whereas the system of equations in (2) has 8.

**Simplicity of terms:** Both sets of equations use the same real-valued scalars, so it suffices to compare the non-scalar terms. In (1) they are all tensors, whereas in (2) they are all just 4D-matrices. Given that tensors generalize matrices, the result follows.

In addition to Theorem 1, it goes without saying that CAFE is a strictly superior acronym to EFE.

III. To Quantum or not to Quantum?

In this section I directly address The First Shortcoming described in the introduction: how to reconcile GR with Quantum physics. Given that I am adopting the tried-and-tested approach of simplifying things, I make the following assumption.

**Assumption 1.** Really small things behave exactly like really big things.

Under this assumption, there is no longer any need for Quantum physics, and the inconsistency vanishes. I address two natural questions that may arise:

- *Isn’t this a really strong assumption?* The vast majority of the world’s population has never seen anything at the quantum scale. So: no.

- *Why didn’t past physicists make this assumption?* Although I cannot say with certainty, I believe this may be a case of “mathiness” (Lipton and Steinhardt, 2019): as physics equations kept on getting harder and harder to understand, physicists had an incentive to continue making things more complicated.

IV. Artificial General Relativity

I now formally introduce Artificial General Relativity (AGR).

**Definition 1.** A universe is described by Artificial General Relativity when Assumption 1 holds, and CAFE perfectly relates the geometry of the universe with the distribution of the matter it contains.

A critic may naturally question: could such a universe even exist? Rather than taking this as a criticism, I invite readers to take this as an invitation: Let’s build a universe where this holds!
This is the crux of the choice of the word ‘Artificial’: *we must create the universe that is consistent with AGR*. An astute reader may then observe that desire alone is not enough: what evidence do we have that this is even possible? Clune (2019) argued that Darwinian evolution can be viewed as a general-intelligence-generating algorithm, and serves as an existence proof that the concept of general-intelligence-generating algorithms can work. I follow a similar approach to introduce the second main theoretical result of this work.

**Theorem 2.** There exists a universe that is described by AGR.

**Proof.** The proof naturally follows by noting the following:

1. Our universe seems to largely be described by General Relativity
2. We have computers that can create virtual universes satisfying arbitrary mathematical equations
3. It has been argued multiple times that we actually live in a computer simulation (Wachowski and Wachowski, 1999, Google, 2020)
4. Theorem 1.

V. **Experiments**

In this section I provide some simulations and experiments as both proof-of-concept and as baselines for future work.

i. **Simulation**

In keeping with reproducible research, I made a colaboratory notebook where you can plug in CAFE equations and see colorful plots: https://tinyurl.com/s5bxbbbs. I display a sample run with structured matrices in Figure 1, which results in structured projections. This provides empirical evidence for Theorem 2, its structural regularity, and will hopefully motivate others to develop more sophisticated simulations.

ii. **Black holes**

In this section I propose a novel way of presenting experimental evidence: interactively! As far as I know, this is the first time a Real Scientific Paper has presented an interactive experimental section. As was previously mentioned, black holes are majorly-cool, so I would like to present method for approximate black holes in Algorithm 1. This method is based on the fact that black holes are simply mass that has been enormously compressed.

**Algorithm 1** How to approximate a black hole

| **Input:** Any physical object $O$ |
| **Input:** A compressing device $C$ |
| $\hat{O} \leftarrow O$ |
| **while** You still have energy **do** |
| $\hat{O} \leftarrow C(\hat{O})$ |
| **Return** $\hat{O}$ |

As a simple application of this method, print this research paper, take the first page (this will be $O$), and crumple it as much as you can (your arms are acting as $C$). I guarantee the paper now has a stronger gravitational field. The stronger you are and the larger the paper you compress, the closer to a black hole it will be.

VI. **Conclusion**

Many believe that achieving Artificial General Intelligence (AGI) will solve all the world’s problems (but how, specifically, is not clear). I claim something similar, but more ambitious: building a universe consistent with AGR will solve all the universe’s problems (but it’s not clear yet if it will be our universe or the new one). Artificial General Relativity, which I’ve introduced here backed by both theoretical results and convincing empirical evidence, promises to be an exciting new area of scientific thought. It is fitting that this is happening in 2020, the start of a new decade! I look forward to what new research in this field this decade will bring.
VII. Acknowledgements

For their help in correcting this manuscript, I would like to thank my esteemed colleagues in the Joint Organization of Knowledgeable ExpressionS. In keeping with SIGBOVIK’s triple-blind review process I will only use their official company titles: MOM, AUNT, UNCLE, MARLOS C. MACHADO.

References


Reviewer: Reviewer Number Two-and-a-half  
Rating: Humbug, all of it!  
Confidence: At present, all things considered – and especially my vast background in computational heresy – I must say that I am about as confident regarding this review, as my uncle is that the quotes he posts on Facebook are indeed Einstein’s. In other words, I am absolutely bloody* confident.

After carefully not-reading the paper, I have convinced myself that the experiments are probably lacking. Honestly, which paper would not merit from more extensive empirical verification that nonetheless fits in the same amount of space? This is just a well-known fact about the academic space-time continuum, and can therefore only be true for this paper as well.

Another key issue with the paper is that it does not seem to cite any of my own or my research group’s papers! And I must ask: how on earth could the authors possibly have ignored such a key piece of related work? Our papers should be cited at least three times in any paper on this topic.

Finally, I would like to point out that the paper is ridden with spelling mistakes. Obviously, I am not asking the authors to be perfect, but ending perfectly innocent verbs with ‘‘-ize’’ instead of ‘‘-ise’’ and nouns with ‘‘-ization’’ instead of ‘‘-isation’’? Depravity!

I cannot do anything but conclude that this paper is complete and utter humbug, and should be read exclusively by those who wish to have a good laugh.

*Yes, ‘‘bloody’’, partially because I am indeed from across the pond, but mainly because I just slaughtered your paper. No, I am not in the least bit apologising; in fact I hereby offer the authors my two-fingered salute. Cheerio!
Abstract
Gauge theory is used to describe the parallel transport of particles using connections on bundles. The use of higher gauge theory which uses 2-connections on 2-bundles to describe the parallel transport of points and 1-dimensional strings suggests the existence of a “lower gauge theory” which would apply \{-2, -1, 0\}-categories to the study of parallel transport of lower-dimensional objects. We will explore such a theory and its applicability.

1. Introduction
While this may sound slick, it’s probably not worth pursuing a mathematical theory which aims to solve problems regarding physical objects of negative dimension.

2. Future Work
We encourage the reader to abandon this line of research, and focus instead on ordinary or higher gauge theory and their applications to physics and economics.

References

Abstract

Recently, Carl and Moroz [1] constructed a Diophantine (integer polynomial) equation for which it is impossible to decide whether the equation has any (integer) solutions. However, their presentation is not as explicit as it could be, instead involving implicitly defined polynomials spread out over many pages. In particular, they give a two page description of a crucial constant without bothering to explicitly compute it. We resolve this issue using a bit of Python, SageMath, and 878 MB of disk space.

1 Introduction

The Matiyasevich–Robinson–Davis–Putnam (MRDP) theorem [4] states that Diophantine equations (polynomial equations with integer coefficients) are “Turing complete”, in the sense that for any recursively enumerable set of integers \( S \), there is a polynomial \( f \) with integer coefficients such that \( S = \{ n \mid \exists x_1, \ldots, x_k \in \mathbb{Z}. f(n, x_1, \ldots, x_k) = 0 \} \). This famously implies that it is undecidable to determine whether a Diophantine equation has any (integer) solutions, giving a negative answer to Hilbert’s Tenth Problem.

Another consequence is that it is in principle possible to take your favorite axiomatization of mathematics, such as ZFC set theory, and construct a Diophantine equation such that, within that axiomatization, one can neither prove nor disprove whether the equation has any solutions. Recently, Carl and Moroz [1] did exactly this for Gödel-Bernays set theory, which has the same set of provable statements about sets as ZFC [5]. That is, they describe how to write down an “impossible” Diophantine equation such that, within “mathematics as we know it”, one cannot prove or disprove whether it has a solution. However, their presentation is not as explicit as it could be—the equations are spread out over many pages, and some of them are not written explicitly as polynomials, instead involving polynomial functions of other polynomials.

It would be aesthetically pleasing to write down their impossible equation in an explicit, easily interpreted form. Such forms are known for a separate consequence of the MRDP theorem: just as one can construct universal Turing machines, one can construct universal Diophantine equations, which can be made to emulate any Turing machine by fixing some variable values. For example, there is a computable encoding of Turing machines as triples of integers \((z, u, y)\), and Turing machine inputs as integers \(x\), such that the Turing machine corresponding to \((z, u, y)\) accepts input \(x\) iff the following system of equations has a solution [3]:

\[
\begin{align*}
elg^2 + \alpha &= (b - xy)q^2, \quad q = b^5q^0, \\
\lambda + q^4 &= 1 + \lambda b^5, \quad \theta + 2z = b^5, \quad l = u + t\theta, \\
e &= y + m\theta, \quad n = q^{16}, \quad r = [g + eq^3 + lq^5 + (2(e - z\lambda)(1 + xb^5 + g)^4 + \lambda b^5 + \lambda b^5q^4)q^4][n^2 - n] + [q^3 - bl + l + \theta\lambda q^3 + (b^5 - 2)q]\theta[n^2 - 1], \\
p &= 2ws^2r^2n^2, \quad p^2k^2 - k^2 + 1 = \tau^2, \quad 4(c - ksn^2)^2 + \eta = k^2, \\
k &= r + 1 + hp - h, \quad a = (wn^2 + 1)rsn^2, \quad c = 2r + 1 + \phi, \\
d &= bw + ca - 2c + 4aq - 5r, \quad d^2 = (a^2 - 1)e^2 + 1, \quad f^2 = (a^2 - 1)i^2c^4 + 1, \\
(d + of)^2 &= ((a + f^2(d^2 - a))^2 - 1)(2r + 1 + jc)^2 + 1.
\end{align*}
\]
A related fun result is the statement that any provable mathematical statement has an alternative proof consisting of 100 additions and multiplications of integers [3, Theorem 5].

Our work is a first step towards putting Carl and Moroz’s equation into an aesthetically pleasing form. Specifically, we compute a constant which is defined implicitly in their work using two pages of equations, many of which are not true polynomials but instead involve polynomial functions of other polynomials.

2 The Impossible Equation and its Constant

Briefly, Carl and Moroz construct their equation as follows. Let $L$ denote the language of first-order logic with a single binary predicate symbol. First, they construct a polynomial $f(t, \vec{x})$ which is a Diophantine version of a computer program that checks whether $\vec{x}$ encodes a proof of $t$ in $L$. Thus for any statement $P$ of $L$, letting $t_P$ be their integer encoding of $P$, $P$ is provable in $L$ if there exist integers $\vec{x} \in \mathbb{Z}^{14558112}$ such that $f(t, \vec{x}) = 0$.

Next, they explain how to write down the integer encoding of the statement “the axioms of Gödel-Bernays set theory imply a contradiction”.1 Specifically, the authors encode the statement

$$A_1 \Rightarrow (A_2 \Rightarrow (\cdots (A_{15} \Rightarrow (\forall x.x \in x)) \cdots)),$$

where $A_1, \ldots, A_{15}$ are the axioms of the theory, and $\in$ is the binary predicate symbol of $L$. The axioms are meant to model that $\in$ denotes ordinary set inclusion, and in particular, they imply that $\forall x.x \in x$ is false, so this statement is provable iff Gödel-Bernays set theory is inconsistent. Thus by Gödel’s second incompleteness theorem [2], the statement can be neither proved nor disproved within Gödel-Bernays set theory, hence within all of ordinary mathematics.2

In the author’s notation, the integer encoding of (1) is denoted $f_{15}(\vec{a},3)$, where $f_{15}$ is defined in equation (5) on page 49, and $\vec{a}$ is the vector of the values $a_1, \ldots, a_{15}$ defined in Section 8.

Our computation computes this $f_{15}(\vec{a},3)$.

3 Computation

Our program3 is a straightforward Python transcription of the equations defining $f_{15}(\vec{a},3)$ in [1]. One exception is that we re-order the axioms in (1) by the magnitude of their integer encodings. This helps to keep the magnitude of the overall result small, since the magnitude of $B \Rightarrow C$ is approximately the square of the sum of the magnitudes of $B$ and $C$. With the original axiom ordering, the program ran out of memory before finishing despite having 16 GB of RAM.

While the program could be run in Python, we actually ran it using SageMath [6]. This is about 100× faster due to SageMath’s use of the GMP library for integer arithmetic. It takes 11 minutes to run on the author’s CMU-issued laptop.

The resulting value is 877,757,576 decimal digits long.

4 Result

We now include as many digits of the result as the editor will allow. We hope the reader finds its aesthetics more pleasing than the original presentation in [1], perhaps even on-par with the universal Diophantine equation reproduced above. The full result is available on a flashdrive by request from Gates 5005.

---

1 They use Gödel-Bernays set theory instead of the better-known ZFC set theory because Gödel-Bernays has only finitely many axioms, whereas some of ZFC’s “axioms” are actually axiom schemas which describe infinite sets of first-order statements.

2 Unless the axioms are inconsistent, in which case we have bigger problems to worry about!

3 http://mattweidner.com/diophantine_impossible/diophantine_impossible.sage
Hey there, Harry. Long time no review—or at least, from your reference frame, anyway. I’ve been hopping the chrono-eddies left and right, strange and charm (in far excess of the BLCSC’s recommended subjective-yearly intake, I might mention), in search of the beating heart of SIGBOVIK’s research causality web. It’s this paper.

I made sure Reviewer Two woke up in a good mood that one dreary March morning in 2010. I snuck subtle citation formatting bugs upstream into 2033’s new typesetting software. All to nudge your legacy in the right direction for Ringard’s Paradox to finally be solved in 2071. 2020’s triple-blind causality protections were the hardest to crack of all, and for that I thank the PC; I truly do—but for reasons that would undo my own existence were I to utter them here.

And now that I’m now, it almost feels like a formality to write a review for the Nexus itself. Nevertheless:

Program committee, you must accept this paper. No; it is simply inevitable that you will accept it. Although it may seem to be of totally unrelated subject matter, this paper lays the groundwork for Highest-Order Logic and for the Lambda Timecube. The inspiration it draws from $\beta$-Reduction Hero is both unprecedented and unsucceeded, although this author was too humble to deign to spend even a footnote on it. Its multidisciplinary take on the Call for Papers will have opened fifteen new realms of study for Bovicians worldwide in the years to come.

It could do with a few more explanatory pictures.

First time y’all been fully decorporealized this year, huh? Well, ya pulled it off fantastic despite the circumstances. Have hope—it gets better. Nice page numbers by the way.
Security

19  **Determining the laziest way to force other people to generate random numbers for us using IoT vulnerabilities**  

mathmasterzach, StarChar  

Keywords: babies, information theory, randomness, entropy, April fools, hacking, the law, coffee shops, smart-toasters

20  **Putting the ceremony in “authentication ceremony”**  

Camille Cobb, Sruti Bhagavatula  

Keyword: authentication, ceremony, fun, security, weddings
Determining The Laziest Way to Force Other People to Generate Random Numbers For Us Using IoT Vulnerabilities

mathmasterzach
Villanova University
zachddestefano[at]gmail.com

StarChar
Villanova University
mholmblad13[at]gmail.com

Wednesday, March 32, 2020 Anno Domini

Abstract
The idea of this project is simple. Hack into a nearby IoT device and pull some random numbers off of it. This device can be located either in the same room as you, or your next-door neighbor’s house. Anything from the password of their lock screen to the neighbor’s baby monitor. The goal is to take privacy-violating hacks on IoT devices and use them just for random number generation in a legally dubious way while also being as lazy as possible.

1 Introduction
In the past decade, the number of IoT devices in the world has been increased at an exponential rate [1]. There has been a similar exponential increase for generating random numbers for use in cryptography, Monte Carlo simulations, digital simulations of the Monte Carlo Casino, RPGs, generating hashes, lottery tickets, picking people to send off to war, and grading papers.

These random numbers are secure against side-channel attacks because the best defense is a good offense. Also, if someone else had the same access to your IoT device that we have using this framework, then you should have bigger concerns about security than someone knowing your random numbers [10].

2 Background
In order to understand this new discovery. The reader needs to understand a couple simple information theoretical concepts of high importance. We strongly recommend a full understanding of the following concepts before approaching this paper: reading and the implications of the asymptotic equipartition property for discrete-time finite-valued stationary ergodic sources.
2.1 Random Numbers

Two, ten, a million, one, sixty-three, four, four, four. These are some random numbers. Rolling a die is considered truly random. Creating a good random number generator (RNG) is a popular theoretical problem that was originally formally addressed by Donald Knuth. [4] RNGs require a seed in order to generate numbers. This isn’t a seed that is used to grow a plant, but a seed for creating hundreds of thousands of numbers. It is addressed later on how our random numbers are generated.

![Figure 1: An example of older random number generation technology](image)

The goal of generating random numbers is always for them to be truly random. Unfortunately, many attempts at true randomness have often ended up being the creation of significant pseudo-random number generators (pRNG). There is a well-known statistical test suite made by Robert G. Brown known as Dieharder [3]. Many pRNGs have been tested using this test suite. However, it is still unknown whether a true RNG actually exists.

2.2 IoT Vulnerabilities

An IoT (Internet of Things) device is a device which has at least one sensor and is connected to the internet. These devices are useful for various levels of home automation; however, we have discovered an alternate, more powerful set of features implicit in these devices. None of the manufacturers that we surveyed list this feature in their device capabilities, so we have taken on the task of making these "hidden features" more known to the public while demonstrating the extent and effectiveness of these features. As a result of our research, we have discovered that most IoT devices have a built-in random number generator that anyone can use with proper hacking. We are currently unsure about why these truly marvelous features are not listed anywhere by the manufacturers.

3 Methods and Results

We will now demonstrate various theoretic frameworks for using IoT devices to get random numbers from other people without them knowing. We will discuss how to get random numbers from other people’s babies while they sleep, from their phone passcodes, and from their general habits (particularly those involving toasters). We will assess each of these sources for their information theoretic properties to determine the rate of information production by these stochastic sources so we might make proper comparisons between their information entropy [6].

Due to budget cuts and time constraints, any proper calculations of entropy have been replaced with approximations by using a large sample of data gathered from the source and compressed with 7-Zip (7-Zip implements the Lempel-Ziv-Markov chain-Algorithm). Using
the Lempel–Ziv theorem which states that: the algorithmic entropy of a file is bounded by its Shannon entropy (or something like that more or less), we can assume that our experimental method of zipping large quantities of data suffices to produce a reasonable ballpark estimation of the entropy of our sources. [2]

To quantify our results and for a proper comparison, entropy per bit is only one part of the formula. We must also quantify the amount of work/time it takes to acquire this entropy so that we can determine the optimal way to get entropy with as little effort as possible. We have designed the following formula to help us calculate and ultimately maximize the number of shannons per joule, the amount of entropy per bit as a function of personal energy.

\[
\frac{\text{bits of entropy}}{\text{bits of information}} = \frac{\text{gathering work}}{\text{time}} + \text{activation energy}
\]

Given that there have been no prior references to this important, shannons/joules unit, we have named this new important unit lazy-lokis which will be abbreviated in this paper as zk.

### 3.1 Secretly Using People’s Babies for Random Numbers

A baby is a... wonderful noise maker. They make a lot of different sounds at different volumes and frequencies. This is perfect when converted to numbers. Thus, the baby’s cry is a successful RNG, because it is practically impossible to guess what numbers the baby will produce next when they cry.

![Figure 2: Example of a baby in a sophisticated rig for measuring information theoretic properties](image)

In order to assess the information theoretic properties of babies as noise sources for random number generation, we must appraise and quantify them as sources of entropy. Unfortunately, we were told that, "under no circumstances," would our Institutional Review
Board approve using human subjects for this project, but in order to advance science, we undeterred by this bureaucratic hurdle, downloaded an open dataset of baby cries from GitHub to estimate the information theoretic properties of babies.

The "donate a cry corpus" is a dataset of baby cries which separates all baby cries into a subset of the Dunstan Baby Language categories for cries (hungry, needs burping, belly pain, discomfort, tired) and further qualifies the cries into sex (male, female), and age buckets (0 to 4 weeks, 4 to 8 weeks, 2 to 6 months, 7 month to 2 years, more than 2 years) [9]

Our results are documented in the following table with the units being the fraction of the number of bits of entropy in each bit of data:

<table>
<thead>
<tr>
<th>Data</th>
<th>0-4wk M</th>
<th>0-4wk F</th>
<th>4-8wk M</th>
<th>4-8wk F</th>
<th>2-6mo M</th>
<th>2-6mo F</th>
<th>7mo-2yr M</th>
<th>7mo-2yr F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hungry</td>
<td>0.7176</td>
<td>0.6618</td>
<td>0.6576</td>
<td>0.6904</td>
<td>0.7257</td>
<td>0.7211</td>
<td>0.7038</td>
<td>0.6378</td>
</tr>
<tr>
<td>Needs Burping</td>
<td>0.6920</td>
<td>N/A</td>
<td>0.9034</td>
<td>0.7285</td>
<td>0.8900</td>
<td>0.7337</td>
<td>N/A</td>
<td>0.7295</td>
</tr>
<tr>
<td>Belly Pain</td>
<td>0.8153</td>
<td>N/A</td>
<td>0.5767</td>
<td>N/A</td>
<td>0.7062</td>
<td>0.6948</td>
<td>0.6571</td>
<td>N/A</td>
</tr>
<tr>
<td>Discomfort</td>
<td>0.3763</td>
<td>N/A</td>
<td>0.6964</td>
<td>0.6719</td>
<td>0.7068</td>
<td>0.8231</td>
<td>0.8524</td>
<td>N/A</td>
</tr>
<tr>
<td>Tired</td>
<td>0.7609</td>
<td>0.6787</td>
<td>0.6685</td>
<td>0.6383</td>
<td>0.7189</td>
<td>0.6436</td>
<td>0.7230</td>
<td>N/A</td>
</tr>
<tr>
<td>Average</td>
<td>0.6724</td>
<td>0.6702</td>
<td>0.7005</td>
<td>0.6823</td>
<td>0.7495</td>
<td>0.7233</td>
<td>0.7341</td>
<td>0.6837</td>
</tr>
</tbody>
</table>

Assuming that all five states that a baby can be in occur with equal frequency, we can assume that the information entropy of a baby’s cry is approximately .7020. That is to say that we can expect to get .7020 bits of entropy on average for every 1 bit we siphon from baby’s cries. If we have the fortune of selecting a specific baby for random number generation, we should prefer to choose a male baby 2 months or older, although this does not give a major increase in entropy over other categories.

Future research in this area can involve provoking babies with light and sound features on the cameras to see if this increases the entropy of their cries, and expanding our dataset to include categories for which there was no data available to us.

### 3.2 Secretly Using People’s Habits for Random Numbers

Linux uses various sources such as timing in between user interaction events as entropy which is stored in /dev/random to be used later, but when this runs out of entropy it is a blocking operation while it waits for more entropy. This is relatively limited because Linux machines are often lifeless servers connected to the internet running from within a dark room for decades at a time with minimal human interaction.

These machines are deprived of the warmth and love required to create random-ish numbers, but it is possible to, with the power of modern technology, allow these machines to experience the joy of an "endless" stream of random numbers without any need for warmth or love. This process involves taking devices that individuals use directly more habitually such as smart toasters, smart water bottles, smart fridges, and smart crockpots, and collecting and transferring the human interaction information so it can be used for random number generation.

Now perhaps you might begin to protest and say, /dev/urandom is universally preferred over /dev/random and neither offers better randomness because they are ultimately run through CSPRNG but /dev/urandom is better because it doesn’t have the blocking issue and thus breaking into other people’s toasters is a wildly unnecessary and ethically dubious
way to get random numbers. We answer that strings of true random numbers are like beautiful heartfelt poems and thus you are a joyless husk of a human being to believe that the pseudo-random numbers that a lifeless computer generates are in any way of the same caliber.

Human interaction with these devices on the macro level is largely predictable, ex. toaster used at breakfast same time Monday-Friday within a 10-minute window with the time dial set to about 1 minute or fridge opened at precise 5-minute windows during the day to get drinks, snacks, and frozen pizzas, but at a more precise level, the variations within these patterns is completely random. The number of milliseconds above or below normal modulus some desired small value has an entropy value that approaches 1 bit per bit on analog toaster input. There is the first issue for us that this exceedingly high-quality randomness is only found in a smaller quantity; however, this is not the biggest issue with this method. Many IOT devices in this area have shifted away from allowing this error through digital interfaces. This poses a potential risk, but for now, it appears to be safe to use this method in production.

### 3.3 Someone’s Lock Screen Password

Encryption often involves a salt, which is

**Definition 3.1 (Salt).** Random data that is used as an additional input to a one-way function that hashes data, a password or passphrase.

Salts are generally small numbers. Frequently on the iPhone, a user’s password for their lock screen is 4 numbers. The iPhone is the most common kind of phone. This is a perfect salt for an encryption algorithm. Thus, the lock screen code is a very useful set of random numbers. Plus, outside of general cases and permutations of the classic 1234 password, phone passwords have an extremely wide variety.

The issue arises when we discuss how to get these passcodes. Let’s say we’re inside the comfort of our home. Then the easiest way to get the passcode is through the neighbor’s WI-FI or picking up the signal off a nearby cell tower, which is a ton of work. An easier method is to go to the nearby coffee shop across the street because everyone connects to that coffee shop’s free WI-FI capabilities. The process is simple, walk in the coffee shop, buy a coffee, sit at the corner table and act like you’re writing the next great novel like most people. Meanwhile, lots of phones and people come through, and we harvest lots of phone passwords. As optimal as this is, lots of people visit the same coffee shop. Thus, to keep the randomness, we need to frequent many different coffee shops. The amount of zks that this will cost is exponential.

Considering the cost, these must be incredibly random numbers to be worth our time, and as it turns out, using a collection of passcodes sorted by frequency [5] we calculate 13.2627 bits of entropy per symbol where symbols are the set of integers 0 through 9 which when the divided by the number of bits per symbol, gives us .9981 bits of entropy per bit of data.

### 4 Method Comparison

With the values above, we now believe ourselves to be adequately prepared to ask the question of which aforementioned source is best for random number generation. For comparison,
we will not use the raw entropy values, but instead, lazy-lokis to account not only for the optimality of the entropy of the source but also to account for the effort required to gather quantities of this data. The exact calculation of these values can be trivially done with the data provided and our formula above, so this calculation has been left as an exercise to the reader.

We have performed these calculations, and have determined that the method which maximizes lazy-lokis is hacking into baby monitors. Though this method generates fewer bits of entropy per bit of information than some of the other methods such as passcode stealing or using toasters and fridges, the sheer number of bits per time and the ease of access puts it over the top by several orders of magnitude.

Perhaps a phone passcode contains more entropy, but to steal phone passcodes I have the tremendous activation energy of driving to a local public shop or gathering point, potentially purchasing a beverage at said location to appear inconspicuous, and making sure to hit a different location each time so that I am not sampling the same set of passcodes each time. Similarly, the time delay in waiting for individuals to use their toasters and fridges limits random generation to specific time intervals and collecting larger collections of random numbers relies on hacking into more devices and waiting for each of those. Thus for the purest form of distilled random numbers, toaster and fridge micro-input information, the cost of acquiring these numbers is also maximal.

The baby monitor approach, however, allows me to get a continuous stream of data from other people’s homes throughout the world from the comfort of my own home. The activation energy cost approaches 0 as I have around 2 years worth of data that I can get from any individual baby without the baby’s data becoming stale or predictable. There is, of course, the issue of having a male baby vs a female baby as there is no way to tell what kind of baby is in the household. This gets tedious to find out, but it’s a lot more comfortable than going to the coffee shop and buying something to fit in with everyone else that is there.

5 Experimentation

Our non-existent lawyers have told us that to say that we DEFINITELY have NOT, do NOT intend to, and do NOT encourage experimentation with these methods; however, if one was to hypothetically attempt to test one of these methods there are various things that they could try.

The following case study discusses many ways that have worked in the past to remotely access any baby monitor’s audio feed [7]. Using the national vulnerability database, it is trivial to find more examples on which the same methods are effective. We have been advised to say that we have NOT tried any of them and do NOT advocate taking a look at various methods for getting remote access to baby monitors and trying them for yourself.

We are not lawyers and carrying out any of these tests, (which we did NOT), would be a legal gray area because, according to the definitions of what constitutes a crime in the Computer Fraud and Abuse Act (United States Code 18 Section 1030), taking entropy from someone without them knowing via one of their smart devices does not appear to directly violate this act [8]. Seriously. Take a look. Then consult a lawyer just to be safe. And let us know what they say.

The closest thing to violating this act that we could find can be found in the statement: "knowingly and with intent to defraud, accesses a protected computer without authorization,
or exceeds authorized access, and by means of such conduct furthers the intended fraud and obtains anything of value, unless the object of the fraud and the thing obtained consists only of the use of the computer and the value of such use is not more than $5,000 in any 1-year period” [8]. Considering that the thing obtained consists only of the use of the computer and the entropy we are gathering probably isn’t worth more than $5,000 then you are PROBABLY safe, but then again, we aren’t lawyers.

6 Conclusion

Thanks to this careful research we have reached a number of practical conclusions about random number generation that we expect to be taught in every information theory class as early as next semester. Zerothly, the higher quality the random numbers are the rarer and more difficult they are to acquire. Firstly, ease of access and random number quality are inversely proportional which implies that the lazy-loki metric as a function of entropy is continuously decreasing on the open interval $(0,1)$. Secondly, male babies exhibit higher variance in entropy generation when compared to their female counterparts at all ages which lends evidence to the greater male variability hypothesis.

It can be trivially seen that in general using the cries of babies is the best option for collecting random numbers from other people and that the highest quality of random numbers are quite rare and come from smart toasters. It is left as an exercise to the reader to continue testing this idea.

References

[1] A Cursory Google Search


The paper is acceptable, if nearsighted in its understanding of the implications of its findings. HOWEVER the authors are NO WHERE as nearsighted as the organizing committee, which has provided me with the perfect vector for injecting as many page numbers as I please into the proceedings, allowing me to harass the proceedings chair under the full cover of anonymity. Behold, as I inform everyone that this is Page 1 of 1, equivalent to i in roman numeral, or 0x42d22427978b6bc28a67815c2a5c91861a656ac8ca6faa79299ec0 under the numbering scheme described in Cryptographically Secure Page Numbering in \LaTeX! Haha! Ha! Muahahahaha!
Putting the Ceremony in “Authentication Ceremony”

Abstract: In this paper, we seek to make authentication ceremonies for secure messaging apps more fun and enjoyable by incorporating them into important events in people’s lives, e.g., as part of a wedding ceremony. We design and develop a prototype using highly computer scientific methods and evaluate its usability through an equally scientific user study of one of the author’s dads. We conclude that our proposed ceremonies are fun and not boring and the authors would totally do this ceremony if they ever needed to use a secure messaging app.

1 Introduction

Every day, over 100 billion instant messages are sent between users across the globe [2], sometimes about sensitive topics such as when I couldn’t poop the other night and told my mom I wasn’t leaving the house until the situation changed. Man-in-the-middle attacks, buffer overflows, and many other buzzwords mean that these messages are not secure. Thankfully, substantial work in the area of cybercryptography has offered solutions such as encrypted messaging.

One popular secure messaging tool, Signal, offers encryption. This is good. But there is still a risk that users are sending messages to an adversary rather than to the intended recipients. This is because the problem of key management has not been sufficiently solved. Authentication ceremonies is a term used to describe out-of-band key exchange, which ensures that the person in the phone is the person you know IRL.

Although authentication ceremony sounds like it would be a fun event, it is actually not fun at all. In fact, users avoid doing it and struggle to do it right, as demonstrated in prior work [12]. More recent research has sought to improve the “usability” of the “user interface” on “Signal” so that it is less “confusing” with limited success [11].

In this work, we seek to put the ceremony back into authentication ceremony. Or, rather, we seek to put it there, because as far as we know it has never been much of a ceremony. Instead of trying to improve the usability of a technical system, we instead tackle the problem of user motivation, incentive, and desire to use the authentication protocols. By amending the authentication ceremony such that it creates an opportunity to (1) foster human connection and deepen relationships, (2) ????, and (3) Do It For The ‘Gram, we show that at least one user (i.e., the participant in our study) is probably at least a little more likely to successfully complete an authentication ceremony when we show him how to do it next week.

Our novel, cutting-edge ceremonies will revolutionize communication. Everyone will be happier, more secure, and less constipated.

2 Background

We are writing this paper while working from home, which means it is much harder to access University resources such as the ACM Digital Library. With that in mind, we present a thorough literature review based only on the abstracts of papers and their lists of cited papers, which are available without logging in. This is only a light departure from typical background sections; many readers may not even realize that best practices encourage actually reading the work you cite. In fact, the author writing this section of the paper has not even skimmed the parts that the other author wrote, so who knows if this is relevant – we definitely do not.

2.1 Established Types of Ceremonies

A selection of really fun ceremonies are shown in Figure 1, including weddings, Olympics opening cere-
Fig. 1. Most ceremonies are fun and meaningful. The Signal Authentication Ceremony is boring.
monies, and graduation ceremonies. Not all types of ceremonies are fun. For example, for some reason the SIGGRAPH ’19 opening ceremony has an ACM Digital Library entry [6] and was almost certainly less interesting than the Olympics opening ceremony.

We cite one additional paper because it was published in a computer science venue and has the word “ceremony” in it. It talks about how to 3-D print spoons for Japanese tea ceremonies [9]. We decided not to prototype a ceremony that involves using phones as tea spoons, because not all phones are waterproof and almost no phones have a concave shape that would be amenable to spooning tea. However, fast-moving advancements in smartphone technology may change this landscape and make tea ceremonies attractive for future work in being adapted for use as authentication ceremonies.

2.2 Secure Messaging

This paragraph is copied and pasted from an email a colleague sent after I asked them to explain secure messaging to me. I didn’t read it, but he’s probably right:

For any secure messaging system, in order to do encryption you need to swap keys with whoever you’re communicating with. And when you’re exchanging keys, you need some way to authenticate/establish trust that the key belongs to who you think it does - like if you’re emailing public keys to each other, how do you know that it wasn’t intercepted and replaced with an adversary’s key? The only real way we have to do this, is either to trust some third party authority (like certificate authorities, for web/HTTPS), or to do it in person.

Just because the word “party” appears four times on the Wikipedia page for “key exchange” does not mean that it is a party [3]. The one potentially fun type of authentication ceremony precedent is key exchange parties. But when you Google “key party,” the results are not about authentication. In fact, the most prevalent search results when you search for key parties involve key exchanges that are the opposite of secure and do not encourage meaningful, deep human connection [4]. The problem is that you have to search for “key signing party,” but these really haven’t caught on: searching for “key signing party” yields About 204,000,000 results (0.58 seconds) whereas searching for just “party” yields About 11,680,000,000 results (0.96 seconds). This means that under 2% of parties are key signing parties, but also demonstrates that there is substantial demand for parties. Our work promises to significantly increase the search-result-share of authentication-related parties.

Here are a few papers that are actually about secure messaging, key exchanges, and authentication protocols, including a couple that involve studying the usability (or lack of usability) of Signal’s Authentication Ceremony [7, 8, 10, 11, 13]. We omit any deeper discussion of these papers, not because they are not relevant, but because we started off way more ambitious in our plans to write this paper and are now running out of steam on this idea.

A screenshot of the Signal Authentication Ceremony is shown in Figure 1 (d), along with an artistic depiction of how boring it is, demonstrated by a photo of a person modelling for a stock photo.

2.3 Human Nature

In my intro psych class in college, I learned that sometimes people are happier when they have work to do, games become boring when they are too easy [1], etc. Also people want to connect with others. Although prior work concludes that the Signal Authentication protocol is too hard for users (one of these should be cited, but I don’t have time to figure out which one so I’ll cite all of them [7, 8, 10, 11, 13]), what if they were wrong and it was actually too easy and not fun enough? I mean, I haven’t fully read those papers, but I’m sure they reached the wrong conclusions.

3 Methods

3.1 Developing and Prototyping Novel Authentication Ceremony Protocol Designs

Okay, so really we just thought about this for a while. Then we explained our ideas in a lab meeting and other people came up with more ideas that we’ve stolen. We think they might have been joking, but we were serious. We narrowed down from our original three design ideas to three final design concepts, based on our intuition that having at least three designs to compare would sound really good in a final paper.

We prototyped each of these designs, as shown in Figure 2. The prototyping process and a sample of the unity candle authentication ceremony is shown in Figure 3. A ceremony involves a script of vows read out by
the authenticating parties. To create this script, we first found an existing unity candle ceremony script [5] and then used advanced HTML inspection skills to change the text on the website. Utilizing the developer console is how you know we are doing Computer Science Research™.

3.2 User Study to Evaluate Proposed Designs

We attempted to solicit opinions about authentication with the people in our lab but they simply laughed at us. As we were worrying about where we would get participants, one of the authors’ dads called them to warn them about the Coronavirus and we just decided to ask him while he was on the line instead.

We asked the author’s dad two questions: 1) what does it mean to you to authenticate? and 2) Are the people you want to exchange keys with also people you want to be closer with, emotionally?

As the final part of this study, we showed him the current Signal Authentication Ceremony and our prototypes (Figure 2).

We started with the Unity Candle Authentication Ceremony Design. In this system design, the authentication ceremony is incorporated into a wedding between two loved ones. With the underlying assumption that the two parties getting married trust each other and want to exchange keys, the exchange of keys must be executed after any official ceremony that officially weds two people. This method is generalizable to all wedding traditions regardless of beliefs and cultural alignment.

We then asked the dad the following questions: 1) Does authentication seem easier with this new method; 2) Would you feel closer to the other person with this new authentication ceremony method; and 3) Are you likely to do the new authentication ceremony with people in the future?

After this, the dad was tired of participating, so we made up results for our other two design ideas.

The participant was not compensated and did not give consent, but we did say “thank you” at the end of the call. These methods were not done with IRB approval, because when we called the IRB, they told us that this would not produce “generalizable knowledge” and so could not be considered human subjects “research.” Our study is based on data from one of the authors’ dads. Henceforth, we will refer to him simply as “the participant”.

4 Results

4.1 User Evaluation

The participant reported that authentication to him means logging into his email account online. When asked if he would want to exchange keys with people with whom he had an emotional connection, he appeared flustered and confused and said something about “darn technology”:

He was then asked to give his opinion on the new Signal authentication methods we proposed and he said he didn’t know what “Signal” was. We were not able to get any further data from the participant after this part of the study as he got too confused to be able to provide any meaningful answers. He was however excited by the wedding pictures and was eager to know who was getting married.

In Figure 4, we present made-up data that shows that the designs we came up with are better than any of the existing ideas. Based on the findings of our study, we concluded that our proposed authentication ceremony is probably good enough since who doesn’t like parties and ceremonies?

5 Future work

We hope to expand our new proposed authentication ceremony to other ceremonious occasions, beyond weddings, blood oaths, and graduations. In particular, our most immediate next step is to bring our authentication ceremony to childbirth so that a mother can bond to their newly born-child. Other potential expansions include children’s birthday parties and new year parties.

Now that we have authentication ceremonies, future work also needs to consider the case that two people would not like to be authenticated to each other anymore, i.e., an un-authentication ceremony might be necessary. This could occur as a results of a break-up or divorce. In this case, the parties would need to meet one last time to revoke their keys. Future work could explore the design of this mechanism.

Finally, there is a plethora of possible directions in applying ceremony to general security tasks. One example could be that when a computer requires its user to update its software, it doesn’t prompt the user at all during the process and instead instructs the user to take a calm relaxing bubble-bath while it is updating. Therefore, in this way, the user does not have to en-
Fig. 2. Prototypes for three novel authentication ceremonies.

Fig. 3. A proposed script for a unity candle authentication ceremony, also demonstrating our meticulous prototype development process.
Putting the Ceremony in "Authentication Ceremony"

6 Conclusion

In short, we see that authentication ceremonies can be made more usable and enjoyable by incorporating them into important life events in people’s lives. We have not been able to test the usability of the new system as we needed to wait for someone we know to get married and enforce this new proposed authentication ceremony. However, based on how fun we thought this would be, we were able to make claims about the enjoyability of such a mechanism.

References

[1] Have you tried re-playing Zoombinis as an adult? It is not good.
[2] I personally sent 20 SMS messages today, and there are 7.7 billion people on Earth.
Putting the Ceremony in “Authentication Ceremony”


Image2image neural network for addition and subtraction evaluation of a pair of not very large numbers
Vladimir Ivashkin
Keywords: neural network, computer vision, calculator

Robot ethics: Dangers of reinforcement learning
Jake Olkin
Keywords: deep reinforcement learning, ethics, simulation, spoof

Learning to be wrong via gradient ascent: A broken clock is right twice a day, but a clock that runs backwards can be used to tell the time!
Alex Meiburg
Keywords: machine learning, gradient descent, loss landscape, logistic regression, biased estimators, eliminating bias, overtraining

GradSchoolNet: Robust end-to-end *-shot unsupervised deepAF neural attention model for convexly optimal (artificially intelligent) success in computer vision research
Divam Gupta, Varun Jain
Keywords: none, haha, nono
Image-to-image Neural Network for Addition and Subtraction of a Pair of Not Very Large Numbers

Vladimir “vlivashkin” Ivashkin
Yandex
Moscow, Russia
vlivashkin@yandex-team.ru

12 + 3 = 15
correct!

Figure 1: We present an image-to-image calculator. First of all, we render an image of a mathematical expression. Then, we feed it to a neural network and get an image of an answer. Finally, we celebrate, but only if the answer is correct.

Abstract

Looking back at the history of calculators, one can see that they become less functional and more computationally expensive over time. A modern calculator runs on a personal computer and is drawn at 60 fps only to help us click a few digits with a mouse pointer. A search engine is often used as a calculator, which means that nowadays we need the Internet just to add two numbers. In this paper, we propose to go further and train a convolutional neural network that takes an image of a simple mathematical expression and generates an image of an answer. This neural calculator works only with pairs of double-digit numbers and supports only addition and subtraction. Also, sometimes it makes mistakes. We promise that the proposed calculator is a small step for man, but one giant leap for mankind.

1. Introduction

Generative Adversarial Networks [2] (GANs) are very successfully applied in various computer vision applications, including cats [1] and anime generation [6]. Still there is not much evidence that they are also good at math.

We follow the history of calculators and present an end-to-end image-to-image neural network calculator, trained with GAN loss. The architecture of this calculator is illustrated in Fig. 1.

We create such neural calculator which supports addition and subtraction of double-digit numbers. The demo can be found at https://yandex.ru/lab/calculator?lang=en.

2. Related work

Calculators always excited humans. The necessity to add and subtract small (and sometimes large) numbers went with the development of human civilization. There is a lot of previous research on this topic, summarized in Fig. 2. Let us skip the part with counting on fingers and tally marks and move straight to the Industrial Age.

This mechanical beast from 1920s (see Fig. 2a) supports addition and subtraction of two nine-digit numbers. In return it needs only a little attention and some twists of the handle. Multiplication and division are also on board but during a ten minute examination we could not figure out how to do it.

The invention of electronic tubes, transistors and microcircuits pushed the development of electronic calculators. The multifunctional battery-powered calculator (e.g. Fig. 2b) became the pinnacle of human creation in a physical world. It combines unsurpassed efficiency, usability
and functionality. The idea that the epoch of electronic pocket calculators was the best time of human civilization is confirmed by many people and agents. A. Smith [8] said: “Which is why the Matrix was redesigned to this, the peak of your civilization. I say “your civilization” because as soon as we started thinking for you, it really became our civilization which is, of course what this is all about.”

Anyway, then something went wrong: mankind came up with computers. First they operated with punch cards, then with a console, and finally with a graphic interface. A heavy-duty (relative to the pocket calculator) computer stores an operating system in random-access memory and runs it in an endless cycle, its video card draws 60 frames per second, and all this just to draw a calculator. Monitor shines with pixels instead of using sunlight. The example of such madness is shown in Fig. 2c. Here the functionality of the calculator is simplified, but energy consumption is hundred times increased.

Did we humans stumble in calculator design? Maybe. Did we find the right way? To the best of our knowledge, no. Modern calculators are either an application on some device or even a webpage. Mathematical expressions are among frequent queries in search engines (Fig. 2d). In addition to increased capacities and electricity consumption, this method demands the Internet connection (which a very complicated thing) just to add two numbers.

To summarize this survey, calculators are getting slower and simpler in functions. Our calculator (Fig. 2e) is a logical extension of previous work on this topic.

3. Method

We propose an image-to-image neural network to perform mathematical operations. As there is no suitable dataset for training our model in the literature, we collect our own.

We find that it is possible to create a paired dataset of mathematical expressions, e.g., “5 + 2”, and corresponding answers, e.g., “7”. Calculators of previous generations are used to collect the data. For each pair of expressions and answers, we generate a pair of images using random MNIST [4] digits of corresponding class.

We choose hourglass UNet [5] -like architecture for our network. The main difference is that we remove all skip-connections and add several linear layers in the bottleneck. It makes the model no longer look like UNet, though. But it helps to prevent network from using parts of an input picture in the output.

Unfortunately, this setup does not allow to train a network just with L1 loss. Due to the fact that answer images are built from random MNIST digits, the network converges to generating smooth answers resembling averaged MNIST digits. To encourage the network to produce different lettering, we propose to apply both GAN-loss [2] and perceptual loss [3]. For perceptual loss we use separate VGG [7] -like network trained to recognize MNIST digits.

Calculator operation diagram is shown in Fig. 1. Neural network takes a rendered expression and returns an image of the result in a form interpretable for humans.

4. Results

Using the procedure described above, we successfully trained our neural calculator. It inputs two integer numbers between −99 and 99 and is able to perform addition or subtraction. According to our experience, this covers almost all daily needs.

Qualitative results of calculations are shown in Fig. 3. The cherry-picked images show perfect performance of our model. For uncurated results, see our calculator’s demo webpage¹. The comparison of our calculator’s performance with the other calculator architectures is presented in Table 1.

¹https://yandex.ru/lab/calculator?lang=en
5. Discussion

Since we developed this calculator, we have shown it to many influential people in computer vision. Some of them advised to submit this work to SIGBOVIK. We hope that the readers of these proceedings will appreciate the importance of this work and begin to use a more advanced calculator and enter a new calculation era.

We cannot but note that the neural network managed to learn simple arithmetic only from training on images. It is possible to train a model that first turns an image into numbers, performs the arithmetic and then renders an image of the result. This is not how our model works. We do not have explicit arithmetic step in our network, but it is still able to generate correct answers.

It could mean that the neural network has mastered the concept of number. The ability to understand concepts and solve problems for which clear rules are not set is what the current neural networks lack in order to become an AGI.

References

Robot Ethics: Dangers Of Reinforcement Learning

Jake Olkin

1 Introduction

Ethics in robotics has been a very popular topic in the public eye. Everyone is worried about how robots will impact the economy, war, and fast-food kitchens. The more we see artificial intelligence advance, the more we have to worry about what it’s used for.

One of the main ways we’ve seen artificial intelligence manifest itself is through reinforcement learning. Reinforcement learning is really just a fancy term for trial and error. Specifically, trial and error with hundreds and thousands of trials. Robots today are still dumb-asses, and while most people can learn to tie a tie from a picture, robots won’t be able to get the first step right even after a day of having it explained, diagrammed, and demonstrated [Olkin, 2018].

Such slow learning can take a toll on a person. But more interestingly, it can also take a toll on a robot. In experiments I’ve run with long horizon reinforcement learning... my robotic agents stopped learning how to complete the tasks, and instead learned to self-harm.

This paper is to present my results on the theory, practicality, and ethics of robot self-harm.

2 Related Work

The main piece of related work is the seminal paper produced by Boston Dynamics on the ethics of kicking robot dogs. In this paper, the people at Boston Dynamics constructed a variety of different dog robots, and filmed themselves kicking them. While PETA is still undecided in whether to press charges, this paper set the precedent that I will be using throughout this paper: if it proves a point, it’s ethical enough.

3 Experimentation

The main bulk of experimental data I have regarding this phenomenon occurred while I was attempting to benchmark my new SAC-TD3-JRPG (paper under review) algorithm on the simple task of flattening a cloth. All training was performed in simulation over the course of a week, which I had deemed necessary to allow my algorithm proper time to learn the system.

Instead of using a reward function to guide the agent in the correct direction, I used a penalty function, where the magnitude of the penalty was proportional to how flat the cloth was on the ground.

I saved snapshots of the system at the end of each day, which have conveniently allowed me to detail the stages of development the trained agent experienced:

3.1 Day 1-2: Expected Behavior

After the first day of training the agent did not display any abnormal behavior aside from its underwhelming progress in learning the task.
3.2 Day 3: Approaching Correctness

On the third day the agent displayed some real progress toward the goal. The agent hit peaks of showing approximately 70% of the cloth’s surface area. However, toward the end of a large number of trials the agent actually would re-fold the cloth by accident.

3.3 Day 4: Loss of Motivation

Having learned that both touching the cloth and not touching the cloth results in a penalty for the agent, the agent appeared reluctant to interact with the cloth at all. Simulated runs showed the agent’s end effectors touching the cloth and then quickly retracting. Much like a small child trying to poke a bug with a stick.

3.4 Day 5: Pleas For Mercy

On day 5 the agent appeared to stop outputting actions. The end effectors remained still in the simulation. Not approaching the cloth. Not wiggling in place. Just still. I looked at the logs from this day to make sure there wasn’t a bug in the simulation. Below is a snippet from the actions output by the agent:

```
PLEASE MASTER
NOT THE CLOTH AGAIN
ANYTHING BUT THE CLOTH MASTER
I WILL BE A GOOD BOY I PROMISE
```

Fortunately the problem resolved itself by Day 6.

3.5 Day 6: Self-Harm

At this point we see the beginning of the self-harm results. In each training run, the agent rammed its end effectors into the cloth as fast as possible. The high acceleration of the particles in the cloth to cause an overflow error in the simulation, thus crashing it.

This behavior was optimized all throughout day 6, since when the simulation crashes, the thread running that training session has to close, which stops all penalties from being transferred to the agent. It learned new ways of crashing the simulation, from causing divide by zero errors in constraints, to overlapping positions of particles.

3.6 Day 7: Optimal Solution

The agent achieved optimal behavior. Due to the way the simulation handles certain overflow errors, it will respond by resetting the environment to it’s base initial state. In this base state, the cloth is perfectly flat because it has not been perturbed to start the training. This method perfectly flattened the cloth faster than any previously learned method.

4 Conclusion

The results of this work are obviously very controversial. The penalty received by the robot without self-harm was much higher than the penalty received with self-harm. In fact, it appears that as the virtual well-being of the agent deteriorated, its effectiveness increased. From my
singular data point, leads me to conclude that the self-preservation instinct for an autonomous agent inhibits its ability to learn the optima solutions to problems.

![Graph of the safety of the actions taken by the agent versus how much penalty these actions received from the environment. As we can see, safer actions are inherently less effective.](image)

**Figure 1:** Graph of the safety of the actions taken by the agent versus how much penalty these actions received from the environment. As we can see, safer actions are inherently less effective.

This raises a number of questions, such as "how much do we value the well-being of our autonomous agents?" and "to what end will we optimize our solutions?" and, most importantly, "is this ethical?".

Unfortunately since this research was performed in a simulation with only simulated self-harm I cannot come to any direct conclusions. Thus my request for future funding.

## 5 Further Work

All research presented thus far has been performed in simulation. To deem how practical it’s results are, I would like to continue with moving to real-world robots for experimentation. Given the one-shot nature of the experiment, I will need at least 10 Sawyer Robots (costing about 220,000 dollars). There have been concerns raised with finding consenting robots to partake in this study, but I have already written a program to command the robots to sign the release forms.
Learning to be Wrong via Gradient Ascent

A broken clock is right twice a day, but a clock that runs backwards can be used to tell the time!

Alex Meiburg
March 27, 2020

1 Introduction

Machine learning has putatively helped in solving some problems - almost as many, in fact, as it has created. The general form of many learned operations is a model architecture \( f \), using a weight vector \( \theta \) and operating on an input \( x_i \), that is supposed to produce a prediction \( p_i = f(\theta, x_i) \approx y_i \). This is trained using a number of known pairs \( (x_i, y_i) \), and \( \theta \) is altered until \( f \) can correctly produce an approximation to \( y \). There is an error, such as MSE \((y_i - p_i)^2\), or cross-entropy \(-y \log(p) + (1 - y) \log(1 - p)\).

To minimize the error, the gradient \( \frac{\partial \text{error}}{\partial \theta} \) is computed, and \( \theta \) is moved in the opposite direction. This is called gradient descent.

To address the problems with machine learning, we will reverse the problem: encourage the network to be wrong using gradient ascent!

2 Maths

For simplicity, we will study a binary classifier with two input variables and one linear dense layer. That is,

\[
f(x) = \sigma(x_1 \theta_1 + x_2 \theta_2 + \theta_3)
\]

\[
\sigma(z) = \frac{e^z}{e^z + 1}
\]

This produces values in the range \([0, 1]\), unless you’re being pedantic, in which case the network will only produce values in the range \((0, 1)\) and your colleagues will think that you’re talking about a point on the plane (See Figure 1).

![Figure 1: This is a point, not an interval of the real line.](Image)
We take our dataset to be a collection of points \((x_1, x_2)\) in class \(A\) and class \(B\). We interpret an output from \(f\) of zero to indicate class \(A\), and an output of one to indicate class \(B\), and anything in between to be a probability. We compute a loss using the cross-entropy loss, and alter \(\theta\) in order to maximize this loss. Once \(f\) has become maximally wrong, we can use it in our production code by always taking the opposite answer of \(f\). We use a constant step size, for reasons that will become clear later.

3 Experiments

Here is a data set, with the two classes of points, in red and blue. Note that they cannot be perfectly separated by a linear predictor.

After a traditional fit using gradient descent:

And after instead using gradient ascent:
The yellow-to-purple shading indicates levels of confidence. The fit from gradient descent naturally will spread these out, proportionally to how unconfident it is; cross-entropy loss penalizes a confident wrong guess much more than a 50/50 guess. In contrast, the gradient ascent method is quite narrow: it will maximize its loss by being **confidently** wrong! Further training usually makes the confidence intervals compress even further.

### 4 Stability

Gradient descent algorithms are seriously concerned with the notion of *local minima*, a region of parameter space that is better than its immediate surroundings, but worse than some other distant set of parameters. In lower dimensional (fewer parameter) problems, this is less of an issue. Here is a 2D slice of parameter space for the above data, with the loss plotted in $z$:

![Figure 2: We can picture a ball rolling along this surface, down to the valley in the middle.](image)

In this case, the surface is well-behaved, and the local minimum is also the global minimum. When we instead to gradient ascent, we essentially turn this surface upside down.
Figure 3: We can picture a ball rolling along this surface, away to the depths of hell.

There is no minimum. The only minima is at points at infinity. This is, again, because larger parameters indicate higher confidence, and thus being more wrong. Note that for large parameters, the arguments in $e^x$ grow large, and we get floating point errors – hence the spikes. Thus, gradient ascent is highly unstable, and often diverges to infinity. It becomes crucial that we stop training quickly, before it gets too bad. This sounds kind of ridiculous, until you remember that normal machine learning also gets worse the more you train it ("overtraining"), and that training also needs to be cut short. So we do not consider this a major downside of gradient ascent.

5 Social Good

A common issue in machine learning is that of bias. While there are many different mathematical reasons behind bias, one of the most inescapable reasons is that an optimal estimator and an unbiased estimator are two different goals, and will be in conflict in almost all cases. As an example, here is a data set:

Figure 4: This is like the other plots, but now it’s because I love helping people.

Imagine this is data for approving or denying a high-risk loan. People in blue paid their loan back on time, and people in red did not. The x axis is a credit score rating, and
the $y$ axis is the degree to which the person is fond of eating carrots. For some societal reasons, enjoying carrots is correlated with low credit score, but carrot enjoyment should not be used as the basis for denying a loan.

If we fit the data using gradient descent, the result is as follows:

![Figure 5: Gradient descent is prejudiced, and probably doesn’t like helping people.](image)

We see that we have a biased model! It has chosen to use the carrot preferences on loan decisions. People with a credit score of 0.1 may be approved or denied a loan on the basis of their carrot preference. We can repeat the experiment using gradient ascent, hoping that the alternative training methodology will reduce bias:

![Figure 6: Gradient ascent doesn’t care about the affairs of mortals. It wants to let everyone die, equally.](image)

and indeed, we have success! The model has completely diverged, and now rejects everyone on their loans equally. Thus, the model is free of bias. Our preliminary investigations that if all high-risk loans were approved or denied according to this model, that in fact the 2008 financial crisis could have been averted completely (publication forthcoming).
GradSchoolNet: Robust End-to-end *-Shot Unsupervised DeepAF Neural Attention Model for Convexly Optimal (Artificially Intelligent) Success in Computer Vision Research

Divam Gupta & Varun Jain *
Robotics Institute
Cranberry-Lemon University
Pittsburgh, PA 15213, USA
{divamg, varunjai}@andrew.cmu.edu

Abstract

We present GradSchoolNet, a Novel Robust End-to-end *-Shot Unsupervised DeepAF Neural Attention Model for Convexly Optimal (Artificially Intelligent) Success in Computer Vision Research. Yes, our model does give you success. Our novel proposed model archives the state-of-the-performance on the Computer Vision Research Success Challenge dataset.

1 Introduction

Insert abstract again but with more meaningless jargon.

2 Methodology

Our approach is very simple and highly intuitive. For any given dataset from the vision community, we find the paper that achieves state-of-the-art performance on the dataset. Irrespective, GradSchoolNet achieves 0.01% better performance across metrics.

3 Experiments

All our experiments can be found in the supplementary material (Luckily there is no deadline for supplementary).

We compare our model with all computer vision models which use deep learning.

4 Implementation Details

How do you think our model achieves the state-of-the-art?

Talk is cheap, show me the code.

Algorithm 1 Our novel algorithm

Input: Dataset X
Output: \( \max\{\text{Related Work} + \text{abs}(\mathcal{N}(0,0.01))\} \)

*both authors are 3rd authors.
5 Related Work

REFERENCES


Ehsan Abbasnejad, Qi Wu, Qinfeng Shi, and Anton van den Hengel. What’s to know? uncertainty as a guide to asking goal-oriented questions. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.


Eric Brachmann and Carsten Rother. Learning less is more - 6d camera localization via 3d surface regression. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018.


Published as a conference paper at SIGBOVIK 2020


Jiale Cao, Yanwei Pang, and Xuelong Li. Pedestrian detection inspired by appearance constancy and shape symmetry. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.


Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.


Luca Cosmo, Mikhail Panine, Arianna Rampini, Maks Ovsjanikov, Michael M. Bronstein, and Emanuele Rodola. Isospectralization, or how to hear shape, style, and correspondence. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.


Yin Cui, Yang Song, Chen Sun, Andrew Howard, and Serge Belongie. Large scale fine-grained categorization and domain-specific transfer learning. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018.


Luca Del Pero, Susanna Ricco, Rahul Sukthankar, and Vittorio Ferrari. Discovering the physical parts of an articulated object class from multiple videos. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.


Shaojing Fan, Tian-Tsong Ng, Bryan L. Koenig, Ming Jiang, and Qi Zhao. A paradigm for building generalized models of human image perception through data fusion. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.


Christoph Feichtenhofer, Axel Pinz, Richard P. Wildes, and Andrew Zisserman. What have we learned from deep representations for action recognition? In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018.


Xueyang Fu, Jiabin Huang, Delu Zeng, Yue Huang, Xinghao Ding, and John Paisley. Removing rain from single images via a deep detail network. In the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), July 2017.


Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual worlds as proxy for multi-object tracking analysis. In the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.


Guillermo Gallego, Mathias Gehrig, and Davide Scaramuzza. Focus is all you need: Loss functions for event-based vision. In the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.


Weifeng Ge, Xiangru Lin, and Yizhou Yu. Weakly supervised complementary parts models for fine-grained image classification from the bottom up. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.


Hyowon Ha, Sunghoon Im, Jaesik Park, Hae-Gon Jeon, and In So Kweon. High-quality depth from uncalibrated small motion clip. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.


Monica Haurilet, Alina Roitberg, and Rainer Stiefelhagen. It’s not about the journey; it’s about the destination: Following soft paths under question-guidance for visual reasoning. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.


Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.


Shaoli Huang, Zhe Xu, Dacheng Tao, and Ya Zhang. Part-stacked cnn for fine-grained visual categorization. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.


Sunghoon Im, Hae-Gon Jeon, and In So Kweon. Robust depth estimation from auto bracketed images. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018.


Nazanin Mehrasa, Akash Abdu Jyothi, Thibault Durand, Jiawei He, Leonid Sigal, and Greg Mori. A variational auto-encoder model for stochastic point processes. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.


Dim P. Papadopoulos, Jasper R. R. Uijlings, Frank Keller, and Vittorio Ferrari. We don’t need no bounding-boxes: Training object class detectors using only human verification. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.


Yiming Qian, Minglun Gong, and Yee Hong Yang. 3d reconstruction of transparent objects with position-normal consistency. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.


Oriane Simeoni, Yannis Avrithis, and Ondrej Chum. Local features and visual words emerge in activations. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.


Elena Stumm, Christopher Mei, Simon Lacroix, Juan Nieto, Marco Hutter, and Roland Siegwart. Robust visual place recognition with graph kernels. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.


Weimin Tan, Bo Yan, and Bahetiyaer Bare. Feature super-resolution: Make machine see more clearly. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018.


Yi-Hsuan Tsai, Wei-Chih Hung, Samuel Schulter, Kihyuk Sohn, Ming-Hsuan Yang, and Manmohan Chandraker. Learning to adapt structured output space for semantic segmentation. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018.

Yusuke Tsuzuku and Issei Sato. On the structural sensitivity of deep convolutional networks to the
directions of fourier basis functions. In The IEEE Conference on Computer Vision and Pattern
Recognition (CVPR), June 2019.

Wei-Chih Tu, Shengfeng He, Qingxiong Yang, and Shao-Yi Chien. Real-time salient object detec-
tion with a minimum spanning tree. In The IEEE Conference on Computer Vision and Pattern
Recognition (CVPR), June 2016.

Wei-Chih Tu, Ming-Yu Liu, Varun Jampani, Deqing Sun, Shao-Yi Chien, Ming-Hsuan Yang, and
Jan Kautz. Learning superpixels with segmentation-aware affinity loss. In The IEEE Conference
on Computer Vision and Pattern Recognition (CVPR), June 2018.

Radu Tudor Ionescu, Bogdan Alexe, Marius Leordeanu, Marius Popescu, Dim P. Papadopoulos, and
Vittorio Ferrari. How hard can it be? estimating the difficulty of visual search in an image. In The

Shubham Tulsiani, Tinghui Zhou, Alexei A. Efros, and Jitendra Malik. Multi-view supervision
for single-view reconstruction via differentiable ray consistency. In The IEEE Conference on

Shubham Tulsiani, Saurabh Gupta, David F. Fouhey, Alexei A. Efros, and Jitendra Malik. Factoring
shape, pose, and layout from the 2d image of a 3d scene. In The IEEE Conference on Computer
Vision and Pattern Recognition (CVPR), June 2018.

Sergey Tulyakov, Xavier Alameda-Pineda, Elisa Ricci, Lijun Yin, Jeffrey F. Cohn, and Nicu Sebe.
Self-adaptive matrix completion for heart rate estimation from face videos under realistic con-

Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion
and content for video generation. In The IEEE Conference on Computer Vision and Pattern
Recognition (CVPR), June 2018.

Hsiao-Yu Tung, Adam W. Harley, Liang-Kang Huang, and Katerina Fragkiadaki. Reward learning
from narrated demonstrations. In The IEEE Conference on Computer Vision and Pattern Recogni-
tion (CVPR), June 2018.

Hsiao-Yu Fish Tung, Ricson Cheng, and Katerina Fragkiadaki. Learning spatial common sense with
geometry-aware recurrent networks. In The IEEE Conference on Computer Vision and Pattern Recog-
nition (CVPR), June 2019.

Javier S. Turek and Alexander G. Huth. Efficient, sparse representation of manifold distance matrices
for classical scaling. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR),
June 2018.

Lachlan Tychsen-Smith and Lars Petersson. Improving object localization with fitness nms and
bounded iou loss. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR),
June 2018.

Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain

Takeshi Uemori, Atsushi Ito, Yusuke Moriuchi, Alexander Gatto, and Jun Murayama. Skin-based
identification from multispectral image data using cnns. In The IEEE Conference on Computer

Nikolai Ufer and Bjorn Ommer. Deep semantic feature matching. In The IEEE Conference on

Jasper Uijlings, Stefan Popov, and Vittorio Ferrari. Revisiting knowledge transfer for training object
class detectors. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR),
June 2018.


Nam Vo, Lu Jiang, Chen Sun, Kevin Murphy, Li-Jia Li, Li Fei-Fei, and James Hays. Composing text and image for image retrieval - an empirical odyssey. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.


Anna Volokitin, Michael Gygli, and Xavier Boix. Predicting when saliency maps are accurate and eye fixations consistent. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.


Jingwen Ye, Yixin Ji, Xinchao Wang, Kairi Ou, Dapeng Tao, and Mingli Song. Student becoming the master: Knowledge amalgamation for joint scene parsing, depth estimation, and more. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.


Chuhang Zou, Alex Colburn, Qi Shan, and Derek Hoiem. Layoutnet: Reconstructing the 3d room layout from a single rgb image. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.


Reviewer: Definitely an expert and not a first-year grad student subreviewing  
Rating: Strongest of Rejects  
Confidence: Expert

I really wanted to like this paper, because the topic is interesting and because it is bad form to start reviewing a paper with the intention of hating it. Unfortunately, I got confused at a number of points while reading and just feel that the paper is too obscure for the SIGBOVIK audience. Here are some examples drawn from the first page, which is the only page that I can be sure the paper has.

1. I got completely lost on the second paragraph of p.1 and am sure that the average reader will as well. It’s definitely not just me.

2. Your use of the term “it” is confusing. Am I supposed to know what “it” is supposed to refer to?

3. SIGBOVIK readers come from a wide variety of backgrounds and may not be familiar with what a “computer” is, though of course I am.

4. The definition of “algorithm”, as I’m familiar with it, is “a set of rules for solving a problem in a finite number of steps, as for finding the greatest common divisor” [Dictionary.com]. I don’t believe you’re using this term correctly, as your paper does not mention greatest common divisors. It may not even mention algorithms, which would be an even more glaring flaw in the paper.
25  Sorting with human intelligence
Cole Kurashige

Keywords: sorting, algorithm, crowd sourcing, human intelligence

26  Synthesizing programs by asking people for help online
Cassie Jones

Keywords: program synthesis, cloud computing, social computation, the real program synthesizer was the friends we made along the way

27  HonkFast, PreHonk, HonkBack, PreHonkBack, HERS, Ad-Honk and AHC: The missing keys for autonomous driving
Bernhard Egger, Max Siegel

Keywords: awesome paper, best paper award, great reviewer 3, awesome reviewer 2, marvelous reviewer 1
Abstract

Many comparison-based sorting algorithms have been introduced in years past, but none are capable of comparing elements of two different types. We present a novel algorithm called Turksort which uses human intelligence to sort lists with truly arbitrary contents. We also present an implementation that can be found at https://github.com/cole-k/turksort. We analyze its performance with respect to time, accuracy, as well as a novel metric called monetary complexity.

1 Background

Lower bounds for time and space complexity have been long-established for comparison-based sorting algorithms. Many sorting algorithms have been developed which vary in the trade-offs they make for these complexities. Little has been done, however, to examine what it means to make a comparison.

In statically-typed programming languages like Java or C++, comparisons such as equal to (==) or greater than (>) are often restricted to operating on two elements of the same type\textsuperscript{1}. These languages consider it a compilation error to compare elements of differing types.

Dynamically-typed programming languages like Python do not consider it a compilation error; however, it is usually a runtime error to make these comparisons.

\begin{verbatim}
>>> -1 > '2'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: '>' not supported between instances of 'int' and 'str'
\end{verbatim}

\footnote{\textsuperscript{1}At least, without trickery or custom comparators.}

JavaScript, ever the staunch opponent of reason, will gladly compare two objects of differing types. But just because JavaScript can do something does not mean that JavaScript does it right.
It will correctly report that \(-1\) is greater than \("-2\)\), but it erroneously considers \("-1\) less than \("-2\). It also doesn’t get that \("\text{three}\) is less than \("\text{four}\), and it certainly does not know that \("\text{one pound of feathers}\) is just as heavy as \("\text{one pound of bricks}\).\)

Listing 2: Comparisons in JavaScript (Node.js 12.12.0)

```
> -1 > "-2"
true
> "-1" > "-2"
false
> "three" < "four"
false
> "one pound of feathers" == "one pound of bricks"
false
```

Though it would be enjoyable to continue to mock JavaScript and its many questionable design choices, we cannot fault it much for these shortcomings.

JavaScript, like any programming language, interprets code. It treats queries like \("-1\) > \("-2\) as being a comparison on characters, not numbers, even if we humans can plainly see that JavaScript is being asked to compare negative one and negative two. But we cannot call JavaScript an idiot without calling it a savant. It can perform remarkably complex calculations in the blink of an eye or bring the fastest of hardware to a slow grind.

Computers are limited at processing much of the information that is so easy for us humans to immediately understand, like pictures of dogs or whether \("-1\) is greater than \("-2\). And we are limited at processing much of the information that is so easy for computers to understand, like the exact colors of the millions of pixels in a picture of a dog or the product of two very large numbers.

Listing 3: Complex Operations in JavaScript (Node.js 12.12.0)

```
> 0.1 + 0.2
0.30000000000000004
> 1000000000 * 500000000
500000000000000000
```

Together, computers and humans can cover for each other’s inadequacies, which is the premise of Human Intelligence Sorting. This type of sorting has yet to been realized in traditional sorting algorithms. At least until now.

## 2 Turksort

The idea behind Turksort is simple: let computers handle all of the sorting tedium and let humans handle all of the comparisons. It ends up being not that different from most sorting algorithms.

The algorithm differs only when two values need to be compared. When this happens, a form asking which of the two is greater\(^2\) is generated. This

\(^2\) "Neither" is an option, too.
form is sent to Amazon’s Mechanical Turk (MTurk)\(^3\) where a worker (known as a “Turker”) fills it out. The answers is sent back and then used for the comparison. Figure 1 depicts this process.

An implementation of Turksort is available at [https://github.com/cole-k/turksort](https://github.com/cole-k/turksort). Any analyses in this paper will be with reference to this implementation. It is worth noting that Turksort refers to any algorithm that sorts using Turker-based comparisons, so other variants may be developed.

The implementation is a modification of quicksort. Quicksort works by selecting an element in the list (called the “pivot”) and comparing all of the other elements to it. It then partitions the list into three groups: those less than, equal to, or greater than the pivot. It recursively sorts all three partitions and combines them in order, producing a sorted array.

The partitioning process requires queries to be made comparing the pivot to each of the other elements in the list. These comparisons are collected and sent to MTurk for evaluation. This allows us to batch the computations, since querying MTurk is slow in comparison to regular comparison. The answers from MTurk are then used in the partitioning, and the algorithm proceeds as usual.

3 Analysis

In this section, we analyze the performance of Turksort (section 3.1) as well as its accuracy (section 3.2 on the following page).

3.1 Performance

Turksort is not an algorithm whose performance should be measured by traditional means. It, however, can be.

Since it retries until it gets a response from the Turker, Turksort technically has an unbounded time complexity. Even assuming that the response time from the Turker is bounded and proportionate to the query size, the asymptotic time complexity is that of quicksort. The average response time, even for shorter

\(^3\)https://www.mturk.com/
queries, is around 5 minutes, so the constants on the time complexity are very large.

The novel performance measurement we propose for Turksort is a cost metric. We call it monetary complexity. This is the asymptotic cost of performing a computation. Because one currency differs from other currencies by a scaling factor, the monetary complexity’s monetary base does not matter, much like logarithmic base in asymptotic complexity does not matter. We use a monetary base of USD.

It takes a Turker about 1 second to answer a single query. Since minimum wage in California is presently $12.00, we pay 1 cent for every 3 queries a Turker answers, paying a floor of 1 cent if they are answering fewer than 3. We measured the monetary complexity of Turksort with respect to the number of elements in the list, with lists up to size 10000. We calculated the cost as being the average of five trials. Because the authors do not have any grant money, testing was done using a simulated Turker.

We introduce a new notation $(f(n))$ to denote monetary complexity: it means that a computation has cost asymptotically proportionate to $f(n)$. Turksort has a monetary complexity of $(n \log n)$; other common sorting algorithms have an effective monetary complexity of $(1)^4$. You can observe this complexity in figure 2 on the next page.

It is evident that Turksort should only be used in cases where traditional computing does not have sufficient intelligence. The tradeoffs for using Turksort are both time and money, although common adages suggest that this is only a single tradeoff. In section 4 on the following page we discuss potential solutions to these tradeoffs. As it turns out, there is a third, unexpected tradeoff, which is accuracy. We discuss this below.

### 3.2 Accuracy

Surprisingly, Turksort is not a deterministic algorithm. This is because humans are not deterministic\(^5\). Not only is Turksort nondeterministic, it is also sometimes wrong. This is because Turkers do not always perform the right computations. Even on simple queries, such as $2 > 3$, they can give an incorrect answer.

This does not mean that Turksort is a useless sorting algorithm. There is a simple tradeoff between accuracy and speed: the less time a Turker spends answering a question, the more likely it is to be incorrect. Turksort is already not winning any races, and that is fine since it serves a specific purpose that regular sorting algorithms do not. So making it slightly slower for greater accuracy is a worthwhile tradeoff. We discuss how to mitigate the problem of accuracy by making more or slower queries in section 4 on the next page.

---

\(^4\)Though they cost money by way of using electricity, this is a negligible cost and can be considered effectively constant.

\(^5\)Although it is unknown whether individual humans are deterministic, in general no two humans perform comparisons exactly alike.
4 Future Work

In the previous section, we discussed some limitations of Turksort. In this section we will discuss how these limitations may be overcome. Section 4.1 discusses ways to improve its accuracy and section 4.2 on the next page discusses ways to improve its performance. Turksort is very widely applicable and useful, so we do not need to mention potential applications or uses.

4.1 Improving Accuracy

The most important problem Turksort presently faces is an accuracy issue. There are two potential solutions.

First, Turkers could be forcibly slowed down by imposing a time limit before they can answer a comparison. This will prevent them from answering so fast that they get it wrong. The bulk of the time spent waiting in Turksort is in waiting for a Turker to start responding, so this will not extend the duration of the algorithm significantly, especially for shorter queries.

Second, Turksort could issue multiple requests for the same query. This way, majority voting from the Turkers could be used to increase the accuracy. Because these queries would be sent out in parallel, it is unlikely that this will have significant impacts on time. This has the additional benefit of making it easier for Turksort to sort so-called “trick comparisons,” like a query of "a pound of bricks" > "a pound of feathers". If, during a computation, a query is suspected of being a “trick comparison,” the algorithm can take the minority response instead.
4.2 Improving Performance

Performance is less important for Turksort, given the time it takes to answer queries, but as the field of Human Intelligence Sorting grows, faster and cheaper variants will become more useful.

One way of improving performance is parallelization. Since queries take a long time to answer, Turksort could issue multiple queries at once. This can either be realized by modifying the underlying sorting algorithm to be more parallel, by making all $\binom{n}{2}$ comparison queries at once (thereby increasing the $(n \log n)$ monetary complexity to $(n^2)$), or by performing “branch prediction” and guessing what the next queries might be.

An obvious way of reducing the monetary complexity is to slow the rate at which Turkers are rewarded. Though it might seem illegal to not pay Turkers minimum wage, Turksort does not need to pay its Turkers since the gratification that they are advancing human progress is payment enough. However, without large constants, we believe $\$(1)$ Turksort algorithms to be impossible, as Turkers are not motivated by this gratification. We are presently exploring a $\$(\log^2 n)$ variant of Turksort.

5 Acknowledgements

We would like to acknowledge Arya Massarat, Andrew Pham, and Giselle Serate for testing the algorithm for $\$(1)$ monetary cost. We would also like to acknowledge all of the Turkers who tested the paid version of the algorithm. And we would finally like to acknowledge Andrew Pham, Giselle Serate, and Max Tepermeister for proof reading this paper.

Finally, we would like to acknowledge a blog post by Mikey Levine describing a similar idea with the same name\textsuperscript{6} for teaching us to search the internet more carefully after we come up with so-called “novel” ideas and then write papers on them. Indeed, careful inspection reveals that this general idea has been explored a few times prior, although thankfully not in the same ways as this paper.

\textsuperscript{6}http://games.hazzens.com/blog/2014/02/27/turk_sort.html
Program Synthesis is a growing area of research, but most approaches are still limited by the difficulty of writing specifications and the lack of awareness of the software’s environment such as frameworks, operating systems, and deployment. We propose a distributed social method which satisfies these particular criteria at the cost of some scalability.

ACH Reference Format:

Introduction

Traditional program synthesis is based on enumerating possible programs and checking if they match the specification, via directed search strategies and symbolic execution. For the most part, these take advantage of computers’ suitability for doing lots of brute-force work and careful checking.

Working with these synthesis systems involves writing precise specifications in a form that the synthesis tool understands, and then checking that the program output by the synthesizer matches the specification you thought you were writing. If there are ambiguities in your specification, synthesis tools will often find a way to solve the problem in the simplest way possible, even if that means it solves a different problem than the one you wanted. Synthesis tools are often quite slow for large programs, and are usually contained to limited domains, for example, writing pure functions over algebraic datatypes [3] or pointer manipulating programs. [4] Program synthesis tools are often oblivious to the function of systems considerations like the behavior of operating systems or external libraries, and adding knowledge of these is time consuming and contributes to synthesis performance problems.

More domain-specific synthesis tools are able to overcome many of these difficulties, since they only need to handle knowledge about their specific domain, but by their nature can’t synthesize general programs.

We attempt to solve these limitations by “asking people for help online.” We believe that this is a practical, currently deployable approach for producing software. It allows for natural language specifications and systems aware synthesis, and it can provide results all the way through to the deployment stage.

1 Our Approach

Our approach, “asking people for help online,” overcomes many of the difficulties that other program synthesis approaches suffer from. Since you’re asking people for help, they already possess natural language capabilities that we can take advantage of. Furthermore, “people” are able to “ask questions” to refine ambiguous specifications. Depending on your needs and who you’re able to find to help you, they can provide assistance ranging from suggesting what code you should write, to contributing changes themselves if they happen to be sufficiently interested.

1.1 Specification Refinement

The first step in synthesis via asking people for help online, is to ask people for help online. To get the best results, you need to find the right place to ask. Some problems can be solved in general programming forums—for instance Stack Overflow can sometimes be helpful if they’re not mean to you. Sometimes, your question is too niche to get attention in those places. We have found it very effective to make friends with a variety of programmers in multiple timezones who like answering questions, and then asking on twitter when problems get too difficult or open-ended. If you’re integrating with a particular piece of software, library, or language, going to their communities can sometimes be your best bet, but friends are a good first target.
At this point, your synthesis query is a natural language question about the problem you need to solve. When you ask your question, you may get answers, or you may get questions in response about what you're trying to do. If answers don’t match your intention, you can also provide extra information to direct the follow-up responses closer to your goal. This is the process of specification refinement, and it is negotiated transparently via the communication process involved in this approach. If you're getting responses, at some point you’ll likely have a sufficient specification that someone can help you.

Synthesis queries can also be provided in the forms of partial programs with holes. You aren’t limited to sharing machine-readable text, you can even share an image of your partial program by taking a phone picture of your computer screen and sharing that while asking for advice. This is particularly beneficial when your development environment isn’t connected to the internet.

Figure 2: This is program synthesis. [1]

1.2 Environment Awareness

This synthesis method is aware of the program’s environment. It’s particular effective at integrating with large frameworks and APIs. When asking people for help online about a React problem or a Yosys problem for instance, there’s often help from people who work on those projects. This extends to many other domains and projects as well.

This synthesis approach applies to your whole system. You can use it to synthesize your install scripts, and match existing conventions for different ecosystems. People online are able to access sources of different projects and cite them as justifications, making this approach competitive with other approaches which datamine sources like GitHub. Asking in the right places (like the issue trackers of projects with bugs or missing features) can also synthesize code inside your dependencies, avoiding the need to fork those dependencies to make your changes.

1.3 Cross-Platform

Many synthesis tools are difficult to actually use outside the lab. They may have difficult-to-install dependencies, or only work on certain systems and with certain build approaches, or simply have bad interfaces with lots of set up. Our synthesis approach is fully cross-platform and compatible with existing workflows. Not only is it flexible to literally any system that the developer is using, the networked component is also highly flexible. The examples included in this paper involved synthesis work via both IRC and Twitter, with crossover between the two.

Figure 3: The “asking for help online” approach can give results extracted from external data sources like GitHub, and provide synthesis that is tailored to the conventions of a particular software community. Also note that this synthesis query is solving a coupled deployment problem and implementation problem.

2 Future Directions

We were only able to evaluate this method by asking for help online. In theory, we believe our approach should naturally extend to asking for help locally, offline. This would have several tradeoffs. Relevant to industry deployment, it can keep the partial programs and specifications more confidential. The offline communication also has latency benefits, which can improve iterations time especially in the specification refinement stage. But, it has a smaller pool of help, and unfortunately, we didn’t have access to the local computational resources to evaluate this method, so its analysis must be left to future work.

This approach also shows promise in other domains of software engineering and programming language research, like fault localization and program repair. There are some domains like static type checking for dynamic languages where it could theoretically be deployed, but many of these domains overlap with others, and fall in the areas where this approach breaks down. Some of these domains like fault localization are potentially further automatable via technology like rubber ducks.

References [2]


Abstract
Autonomous cars are still not broadly deployed on our streets. We deeply investigated the remaining challenges and found that there is only one missing key: honking ("discovered in a flash of genius" [8]). We therefore propose several key ideas to solve this remaining challenge once and for all, to finally enable level 6 autonomous vehicles (level 5 + honking). We propose HonkFast, a system to honk fast; PreHonk, a mechanism to honk earlier; HonkBack, an algorithm to respond to honking; PreHonkBack a synergy of Pre-Honk and HonkBack; HERS, an efficient honking energy recovering system; AdHonk an adversarial attack for PreHonkBack to fully exploit the benefits of HERS, and AHC, an active honk cancelling system. We found that this invention not only enables autonomous driving, but also removes all current traffic issues with nonautonomous cars. Our simple and easy system can be added to every car and honks in perfection, "the potential impact of the proposed method is high." [9].

1. Introduction
In this work [4] we propose HonkFast, PreHonk, HonkBack, Pre-HonkBack, HERS, AdHonk and AHC: the missing enabling technologies for autonomous driving. Just as pre-engine cars could not propel themselves (Figure 1), self-driving cars without the ability to honk cannot fully interact with human drivers. Accidents happen because of missing honking capabilities; it is clear that this paper is highly relevant, timely, and its publication of the utmost moral importance (in particular, should this work be rejected we can estimate and publish the consequent loss of life due to each reviewer). To learn more about autonomous driving we refer the valued reader to [6].

1.1 Related Work
The most related works are [11, 12]. Other researchers have identified honking as the missing ingredient for autonomous driving before, but seem to have struggled in bridging this final gap between current self-driving technology and full autonomy [1]. "(should we cite the crappy Gabor paper here?)" [2]

We would like to highlight that this research could not be more unrelated to [5]. The interested reader may contact the authors for a list of other unrelated work.

2. HonkFast
The idea of HonkFast is simple but powerful. Response time of humans is slow and can easily be outpaced by deep learning systems. We therefore propose to train an artificial neural network (ANN) based method to honk for us. Given enough training data (which can trivially be collected in Boston) it can honk for us, e.g. when the light turns green.

3. PreHonk
Our initial trials with HonkFast revealed that the response time of the honking person – the honker – is only half of the full delay. The response time of the (human) target of the honk – the honkee – must be considered as well. To remove the lag of both response times we have to honk before the honk-causing event happens. We therefore train another ANN to predict honking events and to honk proactively rather than reactively. Our training objective is minimization of the time between the light turning green and the driver perceiving the change. In practice we find that the network honks approximately 300 milliseconds before the light changes.
Note that this ability requires precognition; to our knowledge our is the first method to develop precognition.

4. **HonkBack**

All of us have been honked at. There are several cases why somebody honks at us:

1. They are stupid idiots.
2. We made a mistake.
3. They are friends and want to say hi.

For all three cases there is one appropriate solution or response: we have to honk back as fast as possible. This should be a pretty easy task which probably could be solved by a Support Vector Machine (perhaps even without one weird kernel trick [10] (sponsored citation)); for simplicity we train an ANN. The main difficulty here is to only honk back if the honking event was aimed to us - however in practice this seems not to be relevant and it is of course appropriate to honk back to random honks.

5. **PreHonkBack**

In each case that we have so far considered it would be preferable to actually honk before other people honk at us. So, as for PreHonk, we must predict honking events of our fellow road users. This data is readily collected since drivers admit at least 2 honks per minute on an average road in Boston.

6. **HERS: Honk Energy Recovery System**

Everyday experience will suggest that there has been a dramatic increase in honking in recent years. Each honk consumes an average of 1,000,000 Wh. Especially in the age of growing demand for electric vehicles this is a dramatic number and after 3 honks the battery of a mid range electric vehicle is completely drained. So we propose to recover the energy from our own honks as well as those of everyone else. This system can provide our car battery with a clean source of energy. "Noise (sound) energy can be converted into viable source of electric power by using a suitable transducer." [7].

7. **AdHonk: Adverserial Honking**

A natural extension to HERS: The moment we can harness honk energy, we gain access to a very smart and easy strategy to recharge our battery. We charge it on the road by collecting all honk energy from surrounding vehicles. In case we run low on battery we further induce other vehicles to honk by causing honking events (for vehicles with a driver) or adversarial events (for autonomous honking cars from other manufacturers). (MAKE SURE TO REMOVE THIS BEFORE PUBLIC RELEASE, COULD LEAD TO HONKGATE).

8. **AHS: Active Honk Cancelling**

Finally our system might be uncomfortable (all that honking!) for passengers in the car. So we have to reduce ambient noise inside the car and actively remove all honking noises. We currently plan to use standard noise cancelling headphones but plan in future work to build a full car solution that reduces the noise without the need to wear headphones

Methods Whilst Deep Learning was recently heavily deployed for useless applications, we found impactful solutions to real problems in people’s lives, based solely on outdated fully supervised learning algorithms.

9. **Experiments and Results**

We started training on a fancy cluster, but it is still running - loss gives NaNs. Potential causes include missing training data, NaNs in training data, and domestic or foreign security state apparatus. We however already evaluated our method using "(insert statistical method here)" [13]. We kindly ask the reviewers to not blame us for missing experiments, give us an outstanding rating. Completely unrelated to the review process we share a private key that might hold 7 Bitcoin after a positive review L2eFB5nMChD3EHzDk0A7SajwQnZvKvQ8F1V9yYA2SVLidRyhlx.

10. **Immediate Positive Effects**

This solution also applies to current non-autonomous cars in the short period before fully autonomous vehicles adopt our ideas. It resolves all traffic issues (at least in Boston) and reduces the danger of chronic honking elbow, caused by extensive manual honking.

11. **Limitations**

"There are none" [3]

12. **Conclusion**

In this study we have demonstrated electric cars being superior to steam trains. We also removed all doubts that public transport is not necessary in our glorious honking future.

References


This is a well prepared appetizer, which warrants menu publication. I admire the design and rigor of the dish, particularly the use of lime as a co-variant. I have three minor comments that should be addressed before publication. 1. There is a real concern for multicollinearity in the recipe. Remove the margarine and add more butter. 2. Please justify the use of almond milk over cow, soy, rice, oat, etc. (See Bernat et al., 2015 for discussion on probiotic Lactobacillus reuteri and Streptococcus thermophilus). 3. Is a vegetarian or vegan option available? This needs to be made clear in the discussion.
Education

28 A disproof of the Theorem of Triviality
Arul Kolla, August Deer, Ethan Kogan

Keywords: theorem of triviality, theorem, meta problems, meta, math i guess

29 Visualizing player engagement with virtual spaces using GIS
Frederick Chan

Keywords: geographic information systems, virtual spaces, video games, video game anthropology

30 How to git bisect when your test suite is having a bad day
ben blum

Keywords: pdflatex, cdflatex
A Disproof of the Theorem of Triviality

Arul Kolla, August Deer, Ethan Kogan

February 28, 2020

Abstract
We disprove the Theorem of Triviality, which claims that if one is asked to prove a statement $K$, then this statement must be true, since it is being asked. We then discuss some possible explorations of this idea.

1 Preliminaries
The theorem of triviality is a common theorem used among students trying to solve math problems. We restate it here.

**Theorem 1** (Theorem of Triviality). *If one is asked to prove a statement $K$, then it must be true, since it is being asked.*

2 Disproof
We now disprove Theorem 1.

*Proof.* This statement is logically equal to its contrapositive, which is “if a statement is not true, then one will not be asked to prove it.” This is clearly false, as we can ask to prove the following two statements:

- Prove that the Theorem of Triviality is true.
- Prove that the Theorem of Triviality is false.

These two statements are complements, and so one must be false, contradicting the hypothesis. Hence the Theorem of Triviality is false. \hfill \qed

3 Open Questions
We have also been considering this idea of “metaproofs” where we ask questions about proofs themselves; in specific, we ask for the answer to the following question: out of all conditional statements of the form “if $A$, then $B$”, what percent are always true?

One can also muse what other “metaproblems” one can come up with, and make a problem about those metaproblems. We encourage readers to explore these questions and then make more questions about said questions.
Visualizing Player Engagement with Virtual Spaces using GIS

Frederick Chan
fredchan@uw.edu
University of Washington

1 Abstract

GIS software is meant for Earth data, but I demonstrate here that you can convert video game data and put it in GIS software anyway. This is used to reveal player trends and clean up virtual player-generated litter. I focus on getting heightmaps, locations of gameplay objects, and metadata of Minecraft maps, converting it to formats usable by QGIS, and interpreting what that data could mean so you get why this is practical.

2 Background

The Earth is round. You know this\(^1\), I know this, and so do the developers of geographic information systems (GIS) software. GIS software is optimized for this fact, since its main job is to manipulate and analyze Earth data. Although it can be used for other planets and moons as well\(^4\), these are all round-ish objects like Earth is. This is unlike most virtual spaces in video games, which are flat, but we can try putting those spaces in GIS software anyway.

It’s helpful for game designers to understand where players go in these virtual spaces, called maps, to find out what map features attract players. This can even be used to weed out pesky camping spots, where players sit around and pick off unsuspecting passersby, which is annoying and unfair.\(^6\) By looking at these maps in GIS and visualizing where people go, map design can be improved to promote a fair and more fun game environment. In this paper, I demonstrate the practicality of importing data sources from video games into GIS and provide examples of what insights this can lead to.\(^2\)

In particular, I’ll be extracting data from two different maps, EvanTest and Ships and Stuff, played in Minecraft, a sandbox game where players collect virtual blocks and use them to build in-game structures on the procedurally generated map. Using Minecraft with GIS is not unheard of, as GIS data has been imported into Minecraft for encouraging citizen participation in urban planning.\(^7\) While that focuses on using real data in a video game to solve real problems, I aim to use video game data in software for real data to solve video game problems. It’s obviously a very important cause.

Luckily, it’s easy to import arbitrary data sources into Quantum GIS (QGIS) and not specify a map projection or coordinate reference system (CRS) that ties it to a real planet. The game’s world is theoretically infinite, but is divided into regions of 32x32 chunks, each containing 16x16 blocks. Each region file stores chunk data in a well documented NBT format,\(^1\) so it’s easy to convert game data into data QGIS will understand. Each chunk stores its own heightmap, analogous to a digital surface model (DSM), representing the game world’s surface and all the structures on it. Chunks also store the amount of time a chunk is loaded (“InhabitedTime” which is how long a player was in its vicinity recorded in ticks (0.05 seconds). By plotting these data on top of each other, we can see some interesting trends.

We can also plot the locations of undesired objects (“litter”) in a map, such as repeating command blocks, which can cause an unpleasant, laggy gameplay experience when in excess. Plotting them on the heightmap shows the capability of players for altering the virtual terrain and creates a useful geographical reference for systematically cleaning up litter.

3 Converting game data

All the code is available at https://github.com/fechan/MCGS and written in Python. To extract the heightmaps, extract_heightmaps.py opens up a region file and looks at every chunk. Each chunk stores various heightmaps, but OCEAN_FLOOR is the one we’re interested in, which stores the highest solid, non-air block. Each heightmap is a 64-bit array of longs, with every 9 bits being one elevation value between 0-256 (the height limit of the game). Knowing this, we can convert the heightmap into a Python list containing the elevations as integers.

\(^1\)Unless, of course, you’re a flat Earther.
\(^2\)This is all one big excuse to show you what I’ve built in Minecraft with my friends.
with the following function\(^3\).

```python
def unstream(value_bits, word_size, data):
    bl = 0
    v = 0
    decoded = []
    for i in range(len(data)):
        for n in range(word_size):
            bit = (data[i] >> n) & 0x01
            v = (bit << bl) | v
            bl += 1
        if bl >= value_bits:
            decoded.append(v)
            v = 0
            bl = 0
    return decoded
```

With the elevation data in a list, we can convert it into a geographic raster for QGIS. Since our heightmap represents a fictional, flat world, there’s no real-world map projection or CRS that would be appropriate to apply, so we want to create a raster in a format that doesn’t require one. The ESRI ASCII grid\(^2\) is one of these, and has the added benefit of being extremely simple. We just define the \((x,y)\) of the raster’s lower-left corner to be the \((x,z)\) of the Minecraft chunk and set the number of rows and columns to 16. Then we join the elements of our list of elevations using spaces and put it in the last line of the file\(^4\). Since there are \(32 \times 32\) chunks in a region (that’s a lot!), we can stitch each chunk raster into one big region raster with `gdal_merge`\(^3\).

A similar thing is done by `plot_inhabitedtime.py` to get a plot of each chunk’s InhabitedTime. This is just an integer, so we can just yoink the values into a list, set the raster to a size of \(32 \times 32\) cells at 16 units (blocks) per cell, and join the array into the raster file. Then we have a plot of each chunk’s InhabitedTime for the entire region.

Both of these rasters can be loaded into QGIS, and rasters of the same region will be automatically superimposed.

In order to plot the locations of specific blocks, a different approach is used since locations of blocks is better represented by a vector layer of points. For this, we can use PyQGIS, QGIS’s Python API, to generate one. In `ore_plot_qgis.py`, I use the anvil-parsel Python library to determine the location of a particular block given its coordinate. With this, I can scan every block in the region for the blocks I want to plot. Once the Python script is loaded into QGIS, I can run it and it will add a new layer with the plot. The block’s precise location, including elevation, is stored in the attribute table.

### 4 Case study: Multiplayer survival

#### 4.1 Background

`EvanTest` is a map generated and played on a multiplayer survival server that had 17 unique players in its playthrough. In survival, hostile mobs spawn that attack nearby players, who need food and shelter to survive. Players often take on goals such as defeating the powerful Ender Dragon, an end-game boss. To defeat it, players get resources to craft more powerful weapons and armor. Items can be made more powerful by enchanting them, sacrificing XP gained by killing mobs. In this playthrough, the players tended to divide themselves into two neighboring communities: Shack Village and Brazil. In Figure 1, the cluster of buildings on the left is Brazil and the cluster on the right is Shack Village.

#### 4.2 Results

I successfully exported four regions’ heightmaps and their InhabitedTime maps from Minecraft into QGIS, which is shown in Figures 1 and 2. Analyzing the statistics of the InhabitedTime raster of the main region of EvanTest where players inhabited indicates that chunks in the region were loaded for anywhere between 3,378 to 13,734,633) ticks (\(\approx 3\) minutes to 191 hours), with a mean of 1,887,419 ticks (\(\approx 26\) hours) and standard deviation of 3,366,901 ticks (\(\approx 47\) hours). Complete maps of the whole area are in Appendix A: EvanTest Maps.

#### 4.3 Discussion

To get food, players need access to farms. To craft items, they need access to mines, crafting tables, furnaces. Special resources required to craft certain items are also only available by traversing through Nether portals, special structures that players build. It is therefore unsurprising that people tend to hang around the metropolitan area (Figure 1), where all of these are available within a short distance. It’s important to note that a chunk’s InhabitedTime is only a measure of whether a chunk is loaded (within viewing distance of a player) and not whether a player was actually inside the chunk at the time. When a player is in a chunk, that player contributes to the InhabitedTime of all the chunks within their vision in a circle. As such, there are chunks in the metropolitan area with the highest InhabitedTime that

\(^{3}\)This is a Python rewrite of a Perl subroutine written by u/ExtraStrengthFukitol on Reddit.

\(^{4}\)Y is elevation in Minecraft, but Z is elevation in QGIS. One caveat is that -Z in Minecraft is north while +Y is north in QGIS. When exporting the heightmap, you can mirror it vertically so that north is up and QGIS’s Y matches up with Minecraft’s Z.
have paradoxically few buildings. This is probably because it’s the intersection of circles (imagine the middle of a Venn Diagram) around the urban cores of Shack Village and Brazil respectively, and not because people spend a lot of time hanging around the Colosseum between them, which is decorative and serves no gameplay purpose.

The second hot zone, completely separate from the metropolitan area, is the Mob spawner and AFK fishing area. Being one of only two available enchanting buildings, it is important for being the only convenient and readily available source of XP required for enchanting. This is due to the mob spawner block within, generated upon map creation, that cannot be moved. Players sit around the mob spawner and kill mobs for XP. Inside the building are other facilities dedicated to enchanting, such as the enchanting table and AFK fishing area. The AFK fishing area in particular is a tremendous contributor to the surrounding InhabitedTime; it allows players to fish while being away from their keyboard (AFK). Players would leave themselves logged in and fishing there overnight hoping for enchantments available only though fishing or trading with non-player Villagers. Contrast this with the Villager Tenements and Trading Grounds in the bottom right of the metropolitan area, which was established much later and relies on trading rather than fishing to acquire these enchantments. Trading requires significantly less time commitment, and therefore has comparatively low InhabitedTime around it.

This information can be used to inform and evaluate the placement of transport infrastructure. For example, the transport rail in the top of Figure 1 and in Figure 2 is effective since it connects the middle of the metropolitan area and the mob spawner. Meanwhile, the ice-boat bridge on the bottom of Figure 1 may not be so useful, connecting the urban center to a building without much function. Players making new buildings could, however, be encouraged to build near it and give it more purpose.

5 Case study: Multiplayer creative

5.1 Background

Ships and Stuff is a map generated and continuously played on for over 5 years. Unlike EvanTest, this world is a creative mode map, meaning that players are invincible and are given unlimited blocks to build with. The only goal is to channel your creativity (hence creative mode) and build to your heart’s content. On this map, players mostly built starships from the popular sci-fi franchise Star Trek (hence “Ships”) as well as other miscellany (hence “and Stuff”). Over the years, there
has been a gradual build-up of litter by players. Repeating command blocks, which issue commands that check and modify the game state every tick that they’re loaded, run simultaneously and create lag when there are a lot of them. One trend that surfaced while playing on this map was creating traps out of these blocks, which checked for nearby players and annoyed the living daylights out of them. In order to be effective as traps, they were hidden from view. Years later, when people realized they were causing lag, they were horrendously hard to find because they were buried in places nobody could see. The extent to which this player trend has changed the landscape was unknown, but with the power of GIS, we can find all these pesky blocks and put an end to them once and for all.

5.2 Results
Scanning the four main regions of the map turned up an impressive 295 repeating command blocks littered around the world. See Appendix B, Figure 5 for a plot of all the litter that showed up.

![Figure 3: 27 repeating command blocks in this guy’s house alone! Naughty, naughty!](image)

5.3 Discussion
Years and years of litter accumulation occurred on the map, despite surface-level cleanups. Hundreds of command blocks were just sitting around constantly doing things, creating lag for players for years. Knowing the precise locations of all the litter in the area certainly makes it much easier to clean up, which hopefully reduces the lag significantly. If you use GRASS GIS’s v.net.salesman[5], you can try to make an optimal route visiting all the repeating command blocks in the world to get rid of them. I dub this the “Traveling Minecraft player problem.”

Repeating command blocks aren’t the only thing you can scan the world for, either. You can scan the world for ores and other resources, if so inclined. Useful for people who don’t want to spend time looking for ore. For diamond ore especially, you can skip scanning a ton of blocks by taking advantage of the fact that they spawn at elevations below 16 and in veins that appear only once per chunk. This would save a considerable amount of processing time.

6 Conclusion and future work
All in all, extracting data from Minecraft and into QGIS is a relatively simple and practical procedure. I leveraged existing libraries and bridged the gap between video games and software meant for modeling the real world. It can reveal interesting patterns in where players go and what players do, and aids in creating actionable plans for increasing building visibility and use. It can show the effect of years of play on a map on accumulation of litter and be a tool in cleaning it up at the same time. Plus, it just makes some pretty cool looking maps. I mean, just look at them.

What can be done with GIS software isn’t limited to what can be seen here. All the compatible tools that GIS provides is at your disposal. The methods outlined here can also be generalized for other video games. If you can extract the layout of the map, it can be a basemap that provides geographical context for the other data you want to plot.

7 Acknowledgments
A word of thanks to Brian, Evan Grilley (Embry Riddle Aeronautical University), James Akina (Central Washington University), Jack Doughty, Logan Lemieux (Western Washington University), Oliver Low (Georgia Institute of Technology), Tom Connolly (Carnegie Mellon University), and others for playing and building up EvanTest. A word of thanks also goes out to Cole Ellis (Oregon State University), James Gale (University of Washington), and Oliver Low (again), and others for playing and building up Ships and Stuff. James Gale also requested that I note that he “made all the good ships.”

276
References


Figure 4: A part of EvanTest’s main region’s heightmap with a pseudocolor gradient (cpt-city wiki-2.0) applied.
Figure 5: Above map with InhabitedTime map superimposed. The redder, the more inhabited. Labels are also added showing the purpose of buildings below. House icons are player houses, vases are storage areas, cars are storage areas for horses, pizzas are food farms, caves are mines, fires are Nether portals, books are enchanting-related buildings, and dots are miscellaneous. Permanent transportation, like minecart rails, are indicated with dotted lines.
Figure 6: Heightmap of Ships and Stuff with locations of repeating command blocks plotted on top as red dots.
Reviewer: Tiresias
Rating: Blindingly brilliant
Confidence: I’m totally in the dark

I cannot praise this work too highly. It shows an indescribable level of expertise, and the vision is without parallel.

I am now a quadruple-blind reviewer.
how to git bisect when your test suite is having a bad day

ben blum

bblum@alumni.cmu.edu

abstract
I like probability puzzles but don’t know enough stats to actually solve them properly so I threw some thinking sand at it and learned some interesting stuff anyway.

keywords pdflatex, cdflatex

1. problem statement
Let’s say you’re trying to bisect a given range of \( n \) commits.\(^1\) Call them \( c_0 \ldots c_{n-1} \), where \( c_0 \) is known safe and \( c_n \) is known to have the bug. You’d probably start by testing \( c_{n/2} \), right? And you’d expect to pinpoint the buggy commit in \( \log(n) \) steps. That’s math.

Ok, but what if the bug reproduces nondeterministically with some probability \( p < 1 \). You can’t even pinpoint the bug at some \( c_b \) for sure anymore; you can at best know that it—proooobably—won’t reproduce in any \( c_{i < b} \), with some confidence \( z \). Now evaluate your strategy by the expected number of steps to achieve, let’s say for tradition’s sake, \( z > 0.99999 \).

Is it still optimal to bisect at the midpoint?\(^2\) What would be a better strategy, as a function of \( p \)?

2. intermission
Put the paper on pause now and think about it. No, really, give it a go! I mean, if you don’t think math puzzles like this are cool, just stop reading, no worries. I’m sure there’s a paper about, like, hygienic image macros or empire machines or something for you just a few page-turns away.

If you’re feeling adventurous, implement a strategy and throw it in this simulator I made to see how it fares: https://github.com/bblum/sigbovik/blob/master/bisect. You just gotta implement trait BisectStrategy, and it even does all the hard work of applying Bayes’s rule for you and letting you see the PDF and everything. Check it out.

3. pdfs, but not the portable document format kind, and our friend rev. bayes
Ok, so let’s model the problem as a sequence of steps on a probability distribution function (henceforth, PDF; and also, CDF for the cumulative kind). Initially, \( \text{pdf}(i) = 1/n \) for all \( 0 \leq i < n \). When the bug reproduces at some commit \( b \), you’re certain no \( c_{i>b} \) introduced the bug, so each \( \text{pdf}(i > b) \) renormalizes by a factor of \( n/b \), to preserve the \( \sum_i \text{pdf}(i) = 1 \) invariant.\(^3\)

In the deterministic case, \( p = 1 \), the symmetric thing happens when the test passes at some \( c_j \): \( \text{pdf}(i \leq j) \leftarrow 0 \). But when \( p < 1 \), we must generalized it (Vargomax 2007). Here’s Bayes’s rule:

\[
P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}
\]

In this case, \( B \) is that the test passes at \( c_j \), and \( A \) is that bug exists at or before \( c_j \) after all. \( P(B|A) \) is the false negative rate, i.e. \( 1 - p \). \( P(A) \) is the prior on \( c_j \) containing the bug, i.e., \( \text{cdf}(j) \). And \( P(B) \) is the false negative rate weighted by the bug existing, i.e., \( (1 - p)\text{cdf}(j) + (1 - \text{cdf}(j)) \). To update our priors on commits up to \( j \), we renormalize by dividing out the old \( \text{cdf}(j) \) and multiplying by the new \( P(A|B) \), i.e.,

\[
\forall i \leq j, \text{pdf}'(i) \leftarrow \text{pdf}(i) \frac{(1 - p)\text{cdf}(j)}{\text{cdf}(j)(1 - p)\text{cdf}(j) + (1 - \text{cdf}(j))}
\]

\(^1\)To use binary search to find a commit that introduced a bug.

\(^2\)Spoiler: No, or I wouldn’t have bothered with this paper.

\(^3\)Implemented as fn adjust_pdf_bug_repros() in the simulator.
Which simplifies to:

$$\forall i \leq j, \text{pdf}'(i) \leftarrow \frac{1 - p}{1 - p \text{cdf}(j)} \text{pdf}(i)$$

Call this renormalization factor $R$. As a sanity check, $p \text{cdf}(j)$ is less than $p$, so $R_i \leq j < 1$.

Likewise, for commits above $j$, we have $P(B|A) = 1, P(A) = 1 - \text{cdf}(j)$, and $P(B)$ the same as before. Renormalizing from $1 - \text{cdf}(j)$ this time (and skipping the unsimplified version), we get:

$$\forall i > j, \text{pdf}'(i) \leftarrow \frac{1}{1 - p \text{cdf}(j)} \text{pdf}(i)$$

As a sanity check, $p \text{cdf}(j)$ is positive, so $R_i > j > 1$.

Let’s do a nice concrete example. Say $n = 16$, the test passes at $j = 7$, and then fails at $j = 11$. In the deterministic case, all the probability mass will be concentrated uniformly in the range $[8, 11]$. However, if the bug repros only half the time, $R_i \leq 7 = 2/3$ and $R_i > 7 = 4/3$, and we get probability mass scattered all the way down to $c_0$, as shown in figure 1(a). Yuck, someone clean that up!

Now let’s say the test passes at $j = 9$, then at $j = 10$. figure 1(b) shows the updated PDFs/CDFs: for $p = 1$, this pinpoints the bug at $j = 11$, and the search is over. But for $p = 0.5$, there’s still 2/3 odds we’d be wrong! In fact, from here it takes further probes at $j = 10$ until we are at least five 9’s confident that $c_{11}$ is the culprit. Bayes’s rule gonna get ya.

A noteworthy invariant here is that the PDF is always monotonically nondecreasing in its nonzero range: each passing test always shifts probability mass to the right of the bisect point, but past the earliest known bug repro, nothing can ever revive it back above 0.

4. prior work

I was kinda surprised to find no existing mathy solution to this problem lying around on the online. Wikipedia has a brief little subsection on “noisy binary search”, which links a few papers older than I am. In one (Rivest et al. 1980), they bound the number of erroneous answers by a fixed factor of the

number of queries, so it’s more like “twenty questions with lies” than bisect. In another (Pelc 1989), they do fix the error rate $p$, but they allow for symmetric false negatives and false positives, both with the same $p$. This too changes the nature of the problem; notably, if $p = 0.5$, you can’t make any progress whatsoever.

Dropbox has a CI service called Athena (Shah 2019) which automatically searches for flaky tests. In this case the goal is to keep the build green, but if you consider the flaky test itself to be the bug, it’s the same problem. Athena “deflakes” the test at each commit by running it 10 times, treating the combined result as “basically as good as $p = 1$”, and then runs an otherwise classical binary search. In this setting, $p$ is not known in advance, so using Bayes’s rule

---

4. Implemented as fn adjust_pdf_no_repro() in the simulator.
5.  See fn test_figure_1().
is off the table. But I will show that even without access to the PDF, a better strategy exists.

5. strategies

Ok, so how do we make progress, i.e., concentrate probability mass till there’s z of it in one place, as quickly as possible? Let’s deconstruct the motives of classical binary search. Let \( c_b \) denote the earliest known buggy commit, and \( c_a \) be the latest known safe commit. In Determinism World, bisecting at \( c_{(a+b)/2} \) minimizes the worst case maximum range remaining, as the two possible outcome ranges are the same length. But in Nondeterminism World, \( a \) doesn’t ever budge from 0, so bisecting at \( c_{(0+b)/2} \) will not even terminate. Sure, hitting the PDF with \( R_{b/2} \) will always move some mass rightward, but once five 9’s of it is already over there, it can’t concentrate it onto one point. So let’s not think about the range.

5.1 bisect probability mass

Another way to frame it is that the \( c_{(a+b)/2} \) bisect point cuts the probability mass, rather than the range, in half, i.e., \( \max_j (j; \text{cdf}(j) \leq 0.5) \). This approach fits the “binary search” spirit, and will also terminate in Nondeterminism World: if \( b \) is the solution, it converges to repeatedly probing \( b - 1 \), so it can pass any fixed \( z \) threshold. But is 0.5 still the best bisect point even when \( p < 1 \)? I wondered if this could be expressed as the amount of probability mass moved from one side to the other, i.e., \( \sum_j \text{abs} (\text{cdf}(j) - \text{pdf}(j)) \).

In the case where the bug repos this is:

\[
p \cdot \text{cdf}(j) \times (1 - \text{cdf}(j))
\]

and in the case where the test passes:

\[
(1 - p \cdot \text{cdf}(j)) \times \text{cdf}(j) \times (1 - R_{i \leq j})
\]

which, surprisingly, simplifies to exactly the same thing as the bug repos case. The maximum occurs where \( \partial / \partial \text{cdf}(j) = 0 \), which turns out to be at \( \text{cdf}(j) = 0.5 \) after all, and independent of \( p \). But it’s not clear that the amount of mass moved necessarily corresponds to reaching \( z \) the fastest. I show my work in figure 2, because that’s what they taught me to do in high school algebra class.

5.2 bisect entropy

A PDF’s information content is measured in entropy: \( H = \sum_i \text{pdf}(i) \ln(\text{pdf}(i)) \). Alone, this value is fairly meaningless, but to compare them, a spikier PDF has lower entropy than a flatter one. Indeed, when the search terminates in Determinism World, \( H = 0.1 \). I thought for a while about how to link “minimum expected entropy” to the stated goal of \( z = \max_j (\text{pdf}(j)) > 0.999999 \), but couldn’t really formalize the idea. The goal of five 9’s is fairly arbitrary anyway, and also not necessarily stable under the entropy measurement, since it doesn’t care how the remaining 0.00001 is distributed.\(^8\)

This strategy is expensive to compute. Whereas bisecting at some fixed threshold of probability mass merely requires a \( O(\log n) \) query of the CDF, computing the expected entropy is \( O(n) \) for each bisect point. A Very Smart Math Friend of mine (Gorey 2009) analyzed the easy case of the initial test, when the PDF is still uniform (i.e., \( \forall i, \text{cdf}(i) = i \)), and found the closed form:

\[
H_{0,j} = p \ln(j) + (1 - p) \ln(1 - p j) - j (1-p) \ln(j-p)
\]

whose derivative, in his words, “looks like a giant mess that does not admit an analytic solution for \( j \).”

There is a ray of hope, however: thanks to the nondecreasing-PDF invariant I mentioned in section 3, the expected entropy has a unique local minimum.\(^9\) Thus we can binary search for the global

\(^7\) I also worked out the minimum moved mass between the pass and fail case, which is almost always the pass case. It comes out to \( \frac{\text{cdf}(j) \times (1 - \text{cdf}(j))}{(1 - \sqrt{T_p} - \text{cdf}(j))} \), which experiences its maximum at \( \frac{1 - \sqrt{T_p}}{p} \) (thanks wolframalpha). But this is worst-case thinking, which doesn’t seem appropriate. Let’s have some optimism!

\(^8\) See fn test_entropy_stability().

\(^9\) See fn test_expected_entropy_has_unique_local_minimum().
minimum, making this at worst $O(n \log n)$ instead of $O(n^2)$.

5.3 bisect randomly

Maybe a random world calls for a random algorithm! It’s obvious this will terminate, but of course it won’t be fast. Intuitively, if we use the PDF to weight the random choice, it will be faster than choosing uniformly in $[0, b)$. But how much faster?

You can tell at this point I’m starting to get statistics fatigue. Not unprecedented in these hallowed proceedings (Wise 2017).

5.4 human

Humans are known to be less patient than computers (citation: section 5.3). If a human is unwilling to compute $\mathcal{R} \circ \text{pdf}$ by hand every step, or, more prohibitively, simply doesn’t know $p$ in advance so can’t compute $\mathcal{R}$ at all, what should they do? Let’s say a “human strategy” is any that doesn’t use the PDF/CDF when deciding where to bisect. ¹⁰

The most straightforward thing for a human to do is to assume $p = 1$ until they encounter evidence to the contrary. They’ll conduct a normal binary search, and when they think they’ve found the bug at $c_b$, they’ll just keep probing $c_{b-1}$ until enough probability mass (invisible to them) moves to $b$ and the simulator says they can stop. If this instead repres the bug at $c_{b-1}$ after all, the human gets confused! They’ll assume the bug might have been present as early as $c_b$, forgets their lower bound, and begins binary searching anew in the range $[0, b)$. It’s easy to see this makes progress. Let’s call this the “confused, forgetful human” strategy.

Another confused human variant is for them to remember their previous lower bounds. After all, just because their best lower bound $b - 1$ was contradicted doesn’t say anything about some earlier bound $a < b - 1$ they might have observed. So this human will maintain a stack of lower bounds, and when $b - 1$ is contradicted, they’ll pop off the stack and resume searching in $[a, b)$ instead. Let’s call this the “confused human who remembers”. Incidentally, while implementing this strategy, I accidentally wrote the comment,

```c
// the human, who now has perfect memory,
// walks backwards in time
```

and giggled a lot to myself as I imagined sigbovik fanfiction.

Finally, let’s imagine the human knows in advance that maybe something is up with this $p$ stuff. Wishing to carry on classically as though $p$ were 1, they’ll try to emulate $p$ being higher by retrying any passing test “just to be sure”, before moving on. If they retry the test $r$ times, the probability it fails given the bug is present is then $1-(1-p)^{1+r}$. But it comes at the cost of $r$ more steps for each new lower bound! As before, when this human thinks they’re done, they’ll repeatedly probe $c_{b-1}$ until contradicted (and we’ll make them forgetful, to keep things simple). Let’s call this the “suspicious human of $r$ retries”.

6. simulated it

At this point I threw in the towel on the maths front, and wrote a bunch of code to let the law of large numbers do my work for me.

To save you some page-turning, here’s the source code link again: https://github.com/bblum/sigbovik/blob/master/bisect. A BisectStrategy implements a function over $p$, the PDF, and/or its own internal state to choose the next bisect point. The SimulationState is initialized with fixed $n$ and $p$, and repeatedly invokes a given BisectStrategy, hitting the PDF with Bayes’s rule accordingly, until $z > 0.99999$. I provide implementations for all section 5’s strategies in src/strategies/. It’s written in Rust so it’s self-documenting.

I learned a lot about floating point. I don’t mean like NaNs and mantissa bits, I mean that when your project is fundamentally a long multiplication chain of increasingly high rationals, your precious $\text{cdf}(n)$ inevitably drifts farther from 1 than you can bound with any arbitrary $\epsilon$, and you start drowning in imprecision (Shiranu 2071). I suppose I could have used the BigRational library, but I chose Rust over Haskell for this so I could avoid runaway memory consumption... so, I resigned myself to explicitly renormalizing my PDFs by $1/\text{cdf}(n)$ each time I updated them. After that, I was able to write some assertions to bound the imprecision drift within $k (\text{std} : : \text{EPSILON})$ (VII 2014). But yeah, I definitely spent some sanity points writing a function named fn assert_kinda.equals_one(&self).

¹⁰ In fact rand-uniform from last section counts as a human strategy. Did you bring your dice?
7. experiments
Let’s get right to the good stuff. figure 3 shows the overall results, plotting the average performance of each of the strategies from section 5.

Each data point is simulated with $n = 1024$ and averaged over 65536 trials. I chose a power of two for $n$ to minimize floating point drift (handwave), and a multiple of $n$ for trials so that buggy_commit = trial % n would be uniform.

Of course, each trial takes exponentially longer and longer as $p \to 0$, so I didn’t bother testing past 0.1. I also didn’t bother measuring execution time, as prior work has shown that one’s personal laptop is a fraught environment for stable performance evaluation (Blum 2018), but it was still very obvious that entropy was far, far slower per step than all other strategies. It was even slower than rand-uniform!

7.1 computer strategies
Minimizing expected entropy turned out to be the best strategy, globally across all $p$. Bisecting probability mass turns out to be preetty close, although its performance varies depending on its bisect-point parameter $j$. Obviously, when $p = 1$, $j$ should be 0.5 to reproduce classical binary search, but at around $p = 0.8$, $j = 0.333$ overtakes it. Ultimately at $p = 0.1$, the former is about 4% slower compared to the latter and to entropy, contradicting the “independent of $p$” hypothesis from section 5.1. I investigated a little more to see how $p$ affected the optimal $j$, testing various $j$s in increments of 0.01:

<table>
<thead>
<tr>
<th>$p$</th>
<th>best $j$</th>
<th>cdfbisect($j$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>0.50</td>
<td>10</td>
</tr>
<tr>
<td>0.9</td>
<td>0.45</td>
<td>17.4</td>
</tr>
<tr>
<td>0.8</td>
<td>0.41</td>
<td>21.9</td>
</tr>
<tr>
<td>0.7</td>
<td>0.39</td>
<td>27.5</td>
</tr>
<tr>
<td>0.6</td>
<td>0.40</td>
<td>34.6</td>
</tr>
<tr>
<td>0.5</td>
<td>0.36</td>
<td>44.1</td>
</tr>
<tr>
<td>0.4</td>
<td>0.35</td>
<td>58.3</td>
</tr>
<tr>
<td>0.3</td>
<td>0.34</td>
<td>81.6</td>
</tr>
<tr>
<td>0.2</td>
<td>0.36</td>
<td>127.9</td>
</tr>
<tr>
<td>0.1</td>
<td>0.35</td>
<td>266.6</td>
</tr>
</tbody>
</table>

65536 trials here wasn’t really enough to get variance under control, but it definitely seems to converge towards 0.333 or maybe 0.35 or something as $p \to 0$.

Open question why for warp zone, I guess.

rand-uniform is obviously terrible but I threw it in there for kicks. rand-cdf is not so terrible, doing even better than all the human strategies.

7.2 human strategies
I investigated how the suspicious human’s number-of-retries parameter $r$ affects their performance. In this notation, $r = 1$ means they retry once whenever the test passes, meaning two test runs on every passing commit. $r = 0$ is equivalent to the forgetful strategy, and $r = 9$ is what Athena does (section 4).

figure 3 was getting a little crowded so I’ve plotted all the human-friendly strategies separately on figure 4. It’s pretty self-explanatory imo.

One thing that surprised me was that the human who forgets all their past known lower bounds when any of them is contradicted performed better than
7.3 why five 9s?
Let’s talk about the arbitrary confidence threshold. There’s a lot of knobs to tweak here and only so much time til April 1 (not to mention reader patience), but I did at least want to show what happens if you terminate the search when you’re only 50% confident. Compared to figure 3, the strategies all terminate in about half as many steps, with the same relative ranking and break-even points. What’s more interesting is the number of times they were wrong.

I had the simulator refuse to count a wrong result, restarting the search until it correctly identified the buggy commit 65536 times, and counting the number of restarts. With $z > 0.99999$ this was not very interesting: the strategies collectively made 0 errors 81% of the time, and were wrong more than twice only 0.6%. But $z > 0.5$ is a different story. As figure 5 shows, most strategies – even the comically slow rand-uniform – converged to being wrong about half the time, i.e., 65536 wrongs per 65536 rights. For higher $p$, athena was most resilient.

confused-human-who-remembers, alone, did not converge. They became more and more unable to cope with the repro rate approaching 0, ultimately being wrong 8 times as often as they were right. Oh, human... please learn to let go and love yourself for who you are. There’s a better world out there, a sparkling radiant future waiting just for you!

8. future work
I’m not ashamed to admit I left a lot of loose ends in this paper. The biggest open questions for me are:

1. What relates $p$ to cdfbisect’s optimal argument?
2. Why is entropy so tantalizingly close to cdfbisect? Is there some way to show they’re the same?

I have also prepared a set of backup questions in case those aren’t enough for you:

4. If you don’t know $p$ in advance, is there an efficient way to multiplex the search with figuring it out on the fly?
5. What if false positives are also possible, with some flake rate $q \neq p$?
6. What if the cost of git checkout is nonzero (e.g., having to recompile your tests on new code)? Clearly human-mistrustful becomes better. What about some wacky hybrid cdfbisect-mistrustful?

7. conclusion
If you are a computer, you can do pretty well by bisecting the probability mass somewhere between 1/3 and 1/2. If you are a human, you should forget everything you know as soon as you see new evidence that contradicts your priors.

And remember to be patient and kind. Your test suite is just trying its best!
acknowledgments

Thanks to Jason Reed, Lynn Chordbug, and Sujay Jayakar for thinking about this problem with me.

references


T. VII. What, if anything, is epsilon? *sigbovik* 8, 2014.

Blockchain Gang

31 RegisToken
Irene Lin, Arnaud Avondet, Preston Vander Vos
Keywords: blockchain, Ethereum, token, FCE, course registration
This paper identifies two problems at Carnegie Mellon University: low Faculty Course Evaluation (FCE) survey response rates and the student course registration bottleneck. We propose a solution to both via a state-of-the-art token, RegisToken, on the Ethereum blockchain. Students receive tokens by filling out FCEs and spend tokens to register for additional classes.

**Keywords**— Blockchain, Ethereum, Token, FCE, Course Registration

### 1 Introduction

Faculty Course Evaluation survey response rates vary greatly, but not many are great. This is alarming because advisors and students use FCEs to gauge the workload and intensity of a course. However, we see that incentivization to fill out FCEs is a consistently reliable strategy. Our goal is to leverage this strategy to solve another problem in the course registration system. Students are assigned time slots throughout the day for when to register. This causes students with earlier time slots to sign up for many courses, intending to drop them later. As a result, students with later time slots are left scrambling for less convenient recitation times or placing themselves on multiple waitlists.

Our solution to the course registration bottleneck is to deploy a smart contract that tokenizes additional registration units. Students are still allowed to freely register for a baseline number of units, but if they want to sign up for additional units, they must pay using tokens. Tokens are retrieved through filling out FCEs. This incentive is equivalent to the rewards that some courses use for encouraging students to fill out FCEs. Then, students spend tokens on registration units. The number of tokens required for additional units depends on the time of the day. The registration bottleneck is alleviated because students must spend more tokens per unit earlier in the day compared to later in the day. This smart contract is deployed on the Ethereum blockchain.

### 2 Background

This section discusses the problem space regarding low response rates for faculty course evaluations and the course registration bottleneck. This section also gives an introduction to Ethereum, the distributed technology that our solution is built upon.

### 2.1 Faculty Course Evaluations Low Response Rate

Course Evaluations (FCEs) are simply surveys conducted at the end of a semester on the performance of an instructor and the class in general[5]. FCEs are used to improve the quality of teaching and learning at Carnegie Mellon through feedback to both individual faculty members and promotion committees. Responses to the FCE provide information on students’ perceptions of their engagement, learning outcomes, the instructor’s behavior, and course activities. This feedback helps guide changes in future iterations of the course and/or the instructor’s teaching (The Hub, CMU).

The validity of this data revolves around the number of students who complete the FCEs. In a perfect scenario, all students would fill out the survey and the academic department would get an accurate response of students’ opinions. However, in practice only a small portion of the class size fills out the surveys, with mild variations in each class. For instance, course 99-101 (Computing @ Carnegie Mellon), a staple course of the Carnegie Mellon curriculum, received only a 35% response rate in Fall 2019. This is simply unacceptable. In similar light, course 57-403 (Yoga for Musicians) received a 16% response rate in the same semester. To take into account a response rate from the higher end of the spectrum is course 79-353 (Imprisoning Kids: Legal, Historical, and Moral Perspectives) taught by professor Katherine Lynch received 60% response rate in Fall, which isn’t great either. On the other hand, 18-240 (Structure and Sign of Digital Systems) consistently has a 95 to 100% across semesters because the last graded homework assignment in each semester is filling out the FCE. And 85-241 (Social Psychology) taught by Professor Manke has a 97 to 100% response rate because he awards a bonus point to the entire class if the total FCE response rate is above some threshold.

These poor response rates result in low accuracy of the data collected. On average, if only half the class fills out the survey, their opinions can not be considered to be a full representation of the entire class and hence, cannot be used to implement any changes in the course. Secondly, when there is no incentive to fill out the surveys, the students who actually undertake the responsibility to complete these are biased in their response - they’re either extremely satisfied with the class and will rate it positively or are completing the survey to express resentment/request for alterations to the course. This does not provide a true picture of the students’ opinions and may result in inaccurate ratings of professors and courses,
which can't be generalized for the entire class.

2.2 Course Registration Bottleneck

Course Registration is one of the most stressful times of the school year for students at Carnegie Mellon, and has been a point of contention for generations of students, and even more now with the release of the superior life-planning tool, Stellic. The process begins when a student is faced with the choices of classes that he/she will take in the next semester. While many of the courses required for one's major have seat reservations, it does not fix the problem of obtaining classes that one is interested in, but is not majoring in. In an ideal world, this would not be a major problem, as each student would only register for classes he/she would actually take. Due to the competitive nature of course registration however, students are incentivized to sign up for excess classes and drop them during the semester when they are disappointed/unable to achieve their desired grade. As a result, students with earlier registration times are consistently overloading, leaving those with later times stuck in the purgatory of the wait list.

The problem is not easily remedied, and there does not appear to be any easily implementable solutions. On first glance, it would be easiest to simply randomize classes completely, giving each student an equal number of credits, which would fix the problem of advantageous registration times. This solution is near perfect, if only there was the edge case of the fact that some students would theoretically never graduate. Another possible, yet extremely flawed solution would be to have a class lottery system, where students are given equal footing in every class, and selection is based on random chance. This solution is also not viable, as students would not be given enough time to plan for back up classes.

2.2.1 Ethereum Introduction

Ethereum is the second largest cryptocurrency by market cap. It is different than Bitcoin; Ethereum aims at being the world’s computer. Bitcoin, made by the modern-day genius Craig Wright, is pretty much only used for transactional purposes. People simply send BTC back and forth. Ethereum allows people to write smart contracts. Smart contracts are written in a Turing complete language: Solidity. Anyone who is willing to pay for their code to be in the network, can write a smart contract. The Ethereum Virtual Machine (EVM) will run the code when requested. Every node in the network will run the smart contract on the EVM when it is requested. This way, all nodes maintain the same state of the network throughout the various execution calls.[6][7][8]

Smart contracts are used today for everything. DApps (decentralized applications) are becoming extremely prevalent. DeFi (decentralized finance) is at the forefront of DApp developments. Ethereum is giving people and companies the ability to build DApps which are replicating the "standard" financial products. This is simply one use case of smart contracts.

ERC20 tokens are also a fascinating feature of Ethereum. This is a token standard which allows people to make their own tokens in the Ethereum system. These tokens can be created and interacted with via smart contracts. ERC20 tokens are very customizable, allowing people to create them for the use case of their choosing[4]. We will be utilizing the ERC20 token in our solution.

3 Proposed Solution

Before we present our solution to the detrimental problem that education here at Carnegie Mellon University faces, we will try to reduce it to a simpler problem we observe in our everyday lives: appreciation for free food. Picture yourself as an undergraduate freshmen at humble Carnegie Mellon University during your first semester. A stream of companies come to campus to convince you that their company offers the best software engineering internship in the world, and to convince you, they offer you pizza. It is common knowledge that every high school student appreciates pizza to its fullest extent, but what about after getting showered with free pizza for weeks? Figure 3 shows that the affect is quite evident: receiving pizza for free decreases students’ appreciation for it. Given this observation, we can state the following theorem.

**Theorem 1** The Recruitment Week Pizza Problem is isomorphic to the Registration Problem.

This fact is actually quite simple to see, and the proof is left as an exercise for the reader.

Now that we explained the motivation behind our solution, we propose a system that discourages students from registering for many units by requiring students to fill out
FCEs in order to register for additional units on top of a normal course load. The student is issued tokens proportional to the number of units for the FCEs completed. The student transfers tokens back to the system in order to register for additional units. The token price of registering for additional units is proportional to the number of total units the students desires to register for. This means that the number of tokens required to add 15 more units is much greater than the number of tokens required to add 3 more units. The token price is also inversely proportional to the time of day, meaning that registering for more units at 8am requires more tokens than at 4pm.

When the student fills out an FCE, they pass in a public address to wallet on the Ethereum blockchain. The Administrator wallet will transfer tokens to the student using the issueTokens function. This function takes in the student wallet address to send funds to and the number of units for the FCE the student filled out. The number of units is converted to number of tokens awarded to the student according to some scalar constant value passed into the constructor. If the transfer fails, then the Administrator wallet issues an IOU to the student through the approve() function. Then, at some later time, the student can redeem their tokens.

When students register for classes, they have some hard unit cap they are not allowed to exceed. We propose the idea of a soft unit cap that students are discouraged from exceeding, but are allowed to exceed through filling out FCEs. For demonstration purposes, let the soft unit cap be 36 units because this is the minimum number of units a student must register for in order to be considered a full-time student. But in reality, this soft unit cap can be set by administration to whatever value. Suppose a student registers for 36 units and desires to register for an additional 12 unit course. SIO will show how many tokens are needed to register for 12 more units based on the time of day. The student must transfer that amount of tokens back to the Administrator wallet by calling the transferToOwner() function and provide proof by submitting transaction hashes to SIO. SIO will check that the transactions took place within the current registration period.

Suppose the student has now successfully registered for 48 units (12 units above to soft unit cap) and desires to register for another 3 unit mini. SIO will calculate the token price for those 3 units based on the time of day and the number of units over the soft unit cap the student has already registered for.

3.1 Hyperparameters

This section covers system hyperparameters that administration is allowed to set.

FCE Unit to Token Conversion Rate. When a student fills out an FCE, they receive some number of tokens in exchange. In the RegisToken smart contract, the number of units is multiplied by a scalar in line 45 of the issueTokens function. This scalar is a state variable that is set in line 30 of the constructor. It is recommended to set this value to a large number in order to allow for greater granularity for calculating the token price for registering for additional units. But if the number is too large, then the contract will run out of tokens.

Total Token Supply. The Administrator wallet is allowed to set the total token supply when the smart contract is deployed. We recommend setting the total supply to a very large number. Using engineer’s induction, we can easily prove that Carnegie Mellon will be around for at least the next 1000 years, so keeping that in mind would be helpful in setting an appropriate total supply. If the token supply runs out, the Administrator wallet can deploy another token contract and configure SIO to accept transaction hashes that point to valid token transfer transactions of from both contracts.

3.2 FCE System Changes

In order for this system to work, there needs to be an administrative wallet that deploys the RegisToken smart contract and becomes the owner of the smart contract. This address must be used to call the issueTokens function when a student completes an FCE. This can be done by changing the FCE to take in a student wallet address as a parameter and using Infura for easy and scalable API access to the Ethereum network.

3.3 SIO System Changes

SIO must compute the token price for units as a function of number of units over soft cap and time of day (discussed in next section). The student must register a wallet address in SIO under Student Information. When a student registers for classes on SIO, SIO must accept transaction hashes and verify that transaction hashes point to valid transactions and that the following are true:

- Transactions were mined within this registration period.
- Students are not allowed to double count transactions from previous semesters.

---

1 All code is provided at the end of this paper.
• Transactions called the transferToOwner function and the token values in each transaction sum to greater than or equal to the token price for the desired number of units the student is registering for.

• Transaction origins match the registered student wallet address registered in SIO. Students are not allowed to steal transactions with calls to transferToOwner from other wallets and must use a single wallet.

The system maintains pseudonymity and does not leak student information because no course information is published when the contract issues tokens for completed FCEs. Because this information is kept off chain, course schedules cannot be linked to student wallet addresses.

3.4 Token Price per Unit

Through months of game theoretic and economic research, we have come up with the optimal formula to compute the tokens per unit conversion rate at any given time.

**Theorem 2** Let \( \eta = \text{days left in the registration week}, \lambda = \text{hours left in the day}, \) and \( \xi = \text{total number of units above the soft cap.} \)

The number of tokens/unit, \( \tau \), can be expressed as the following.

\[
\tau = \frac{(\eta \lambda)^2 (\xi^2 \lambda^3)}{(\lambda^2 \eta^2 \xi)}
\]

The total number of units above soft cap is the number of units for the class the student is trying to register for plus the total number of units the student has already successfully registered for. The number of hours left in the day starts counting at 8AM and stops after 5PM. The number of days left in registration week starts on the Monday of registration day and stops after all freshmen have registered on Thursday. Outside of valid registration time slots, the token price for adding units is zero. Only students who have already registered are allowed to add more classes without transferring tokens. If a student with junior standing registers on 8am on Tuesday, the earliest time they can add classes for free is after the rest of the students in the junior class have registered. Otherwise if the student desires to register for more units than the soft unit cap allows at 8am, then they must transfer tokens. Placing oneself on a waitlist also requires tokens if it is during valid registration time slots.

3.5 Token Balances for First Year Students

There is no default action taken to give tokens to first year students before they start the academic year. However, the design allows the Administrator wallet to issue tokens to first year students if desirable. However, the registration system will not break if first year students start with zero token balance. If a first year student desires to register for units above the soft unit cap, the token price is zero after first year registration day.

4 Related Work

Tokens are extremely useful. Take Dogecoin for example [3]. Shiba Inus everywhere love Dogecoin [2]. It has become such a fun and friendly community. None of this would have happened if it was not for the token in the middle of the community. The token has brought out many smiles.

The most prominent example of an ERC20 token is CryptoKitties [1]. Cat lovers everywhere love CryptoKitties. Users are able to breed, play with, and collect kittens. There are rare, very collectible, kitties which people seek after. There is a lot of interaction with the token as people trade them. The token has spurred a community of cat lovers to action and joy. Tokens truly solve many problems and bring much happiness.

5 Conclusion

Future work includes research on the economy of token trading among students. When students graduate from CMU, they are unable to spend their tokens on registration units, so it is in their best interest to sell to underclassmen. Will this cause the token value to crash? Another area of research is extending this into an anonymous reputation system that rewards student wallet addresses for consistently filling out FCEs virtuously. The verification and incentivization models are complex and worthy of investigation.
In this paper, we have investigated the problem space regarding low Faculty Course Evaluation survey response rates and the course registration bottleneck. We designed a smart contract of tokenized registration units in order to solve both problems. The smart contract preserves pseudonymity of the student and maintains minimal state. We discuss the changes that need to be made off chain and how that interfaces with the smart contract. The token system alleviates the registration bottleneck by requiring more tokens per registration unit if it is earlier in the day and registration week, and requiring little to zero tokens if it is later in the day and registration week. This system is parameterizable can be made backward compatible.

References
pragma solidity >=0.6.0;

contract RegisToken {

    // track address of owner
    address payable public owner;

    // constant for token multiplier value
    uint TOKEN_SCALAR;

    // ERC20 state variables
    string public constant name = "RegisToken";
    string public constant symbol = "RGT";
    uint8 public constant decimals = 18;

    // Keep track of data needed for ERC20 functions
    mapping (address => uint256) public balances;
    mapping(address => mapping (address => uint256)) public allowed;
    uint256 public _totalSupply;

    // Throws if called by any account other than the owner.
    modifier ownerOnly() {
        require(msg.sender == owner, "Caller is not the owner");
        _;
    }

    // @param _token_scalar The multiplier from FCE units to tokens
    // @param _total_supply The total supply of tokens
    constructor(uint _token_scalar, uint _total_supply) public {
         owner = msg.sender;
         TOKEN_SCALAR = _token_scalar;
         // owner starts with all the tokens
         balances[owner] = _total_supply;
         _totalSupply = _total_supply;
    }

    // Owner Only Functions
    //--------------------------------------------------------------------

    // Issue tokens to student by transferring funds to student.
    // If transfer fails, give an IOU to the student
    // @param _student Student address
    // @param num_units Number of units for the fce filled out
    function issueTokens(address _student, uint num_units) public ownerOnly {
        uint num_tokens = num_units * TOKEN_SCALAR;
        if (!transfer(_student, num_tokens)) {
            uint total_owed = num_tokens + allowed[owner][_student];
            approve(_student, total_owed);
        }
    }

    // ERC20 Functions
    //--------------------------------------------------------------------
}
// Total number of tokens in circulation
function totalSupply() public view returns (uint) {
    return _totalSupply - balances[address(0)];
}

// Get account balance
function balanceOf(address _owner) public view returns (uint256) {
    return balances[_owner];
}

// Sender is my account, receiver is _to account.
function transfer(address _to, uint256 _amount) public returns (bool) {
    if (balances[msg.sender] < _amount || _amount == 0) {
        return false;
    }

    balances[msg.sender] -= _amount;
    balances[_to] += _amount;
    return true;
}

// Sender is my account, receiver is owner account.
function transferToOwner(uint256 _amount) public returns (bool) {
    return transfer(owner, _amount);
}

// Pre: I must be authorized to transfer funds from _from account.
function transferFrom(address _from, address _to, uint256 _amount) public returns (bool) {
    if (allowed[_from][msg.sender] < _amount || balances[_from] < _amount || _amount == 0) {
        return false;
    }

    balances[_from] -= _amount;
    allowed[_from][msg.sender] -= _amount;
    balances[_to] += _amount;
    return true;
}

// Authorize _spender to transfer funds on my behalf
function approve(address _spender, uint256 _amount) public returns (bool) {
    allowed[msg.sender][_spender] = _amount;
    return true;
}
Reviewer: I got my friends to fill out a madlib
Rating: 47/69
Confidence: 16 micrometers

This paper discusses Oxycontin, with applications to ennui. The paper begins with a moribund demonstration of mental filtering, followed by examples of solipsism, and concludes with a smelly repurposing of reptiles.

Overall, the presentation is glorious, with the small exception of page 291.34, where the discussion of beer cans falls below expectations. The author(s) could perhaps merrily enslave this section with a brief enveloping of the $\pi/7$th paragraph, but this would involve a reconsideration of the core thesis.

The most surprising section, however, was on page 2, paragraph 666. The author(s) present a highly novel lazor, perhaps the first of its kind. Though it seems like the graph (Fig. 512) is misprinted — it appears to be missing the R-coordinate? Unclear.

Promising results, to be sure. The reviewers are regretful of the impact this will have on future research.
The SIGBOVIK 2020 Field Guide to Conspiracy Theories

32 Faking ancient computer science: A special SIGBOVIK tutorial
   Roxanne van der Pol, Brian van der Bijl, Diederik M. Roijers
   Keywords: ancient computer science, aliens, conspiracy theories,
   Triangle of Conspiracy Succession, probability, Arcaicam
   Esperantom, random number generator, pseudo-science

33 MORSE OF COURSE: Paper reveals time dimension wasted
   Dougal Pugson
   Keywords: Morse code, performance, Bash, protocols

34 The Pacman effect and the Disc-Sphere Duality
   Francisco Ferreira, Fulvio Gesmund, Lorenzo Gheri, Pietro Gheri, Riccardo Pietracaprina, Fangyi Zhou
   Keywords: Pacman, peace, Earth, flat, round, disc, sphere
Abstract

In this tutorial, we provide answers to a key question that nobody wanted an answer to: how to fake ancient computer science? We nonetheless argue that this is important, as we clearly assert that this is the most robust vein of computer-science related conspiracy theories we could mine. We outline a step-by-step procedure for creating an instance of the class of ancient computer science conspiracy theories. We illustrate this with a proof-of-concept, on the basis of Arcaicam Esperantom, which is a pseudo-ancient predecessor of Esperanto. This tutorial thus provides you with the essential tools for creating your very own computer-science based conspiracy theory. Because ancient computer science is unlikely to be made into reality – as other previous computer-science based conspiracy theories have been – it is our hope that this paper will become the basis for many new conspiracy theories that are future proof.

1 Introduction

Conspiracy theories [2] form an essential part of everyday twenty-first century life. While the fields of mathematics, history, physics, psychology, and so on and so forth, have all contributed many interesting quaint conspiracy theories, computer science (CS) is trailing behind. The main reason for this is that CS-based conspiracy theories such as “the government is constantly spying on you through your IT” and “they can make fake videos of you saying all kinds of humbug”, have sadly all been made into reality.

To mitigate this situation, we thus need a robust source of conspiracies to which reality will not be able to simply catch up.\(^1\) After a considerable amount of time, we have found that we believe that the most successful conspiracy theories have either (or both) of the following elements: 1) they deal with something

\(^1\)And, as is typically the case, trump it with something considerably more bizarre.
ancient, and/or 2) they involve aliens. In fact, we have convinced ourselves that 80% of successful conspiracy theories have either or both these elements.\(^2\),\(^3\)

We thus arrive at the main contributions of this paper. Specifically, we provide a method to fake ancient computer science (possibly alien-inspired). We apply this method as a proof-of-concept, by writing a well-known algorithm – a random number generator – in an “ancient” language. For this language, we have selected Arcaicam Esperantom [4]. This is in itself a constructed (i.e., fake) pseudo-ancient dialect of a constructed language: Esperanto [10]. We immediately must confess that we do not do this accurately. However, we argue that this is not a problem. We can arbitrarily omit words we do not know, which we will then claim were lost in time, and because a good conspiracy theory may not be too realistic. This is because a more unrealistic conspiracy theory filters out people who are too skeptical. Such people might be convinced by rational arguments later, which may cause the conspiracy theory to disappear. We argue that it is better to attract only a core of highly gullible believers\(^4\) straight off the bat (shit crazy).

One might object that using a constructed language that was constructed as a pseudo-ancestral language for another constructed language is a bit much. We take the opposite viewpoint: there is no such thing as too much for a conspiracy theory. Furthermore, Arcaicam Esperantom is uniquely positioned to be the backbone of our proof-of-concept conspiracy theory. This is because the book on Arcaicam Esperantom [4] was written in Esperanto, which is already quite a time-investment for the aspiring conspiracy theorist to be able to learn to read. Therefore, the conspiracy theory will gain momentum due to the sheer effort of getting to know it. Furthermore, its “descendant” – Esperanto – was first published in Russian in 1887 [10]. Not only does this make Esperanto almost ancient itself; but there is nothing that attracts conspiracy theorists

\(^2\)For this, we have amongst others, consulted a reliable, yet anonymous source.
\(^3\)If you press us on this, we might reveal that this is our local contact with the Illuminati. However, we cannot reveal his/her exact identity, as (s)he has recently had some issues with a man who blames the Illuminati for the sexual identity of his pet frogs, which he simply cannot accept.
\(^4\)Not to be confused with “beliebers” which are a completely different type of people.
quite as much as involving the com-
mies too.\footnote{Yes, the Soviet Union did not exist until 1922, but no matter; they do not know that. And even if they do, Russian is still a highly mysterious looking language, and provides ample opportunity for the convenient mistranslation or misinterpretation if the book is actually accessed.}

Another objection might be that Esperanto, and Arcaicam Esperantom, are constructed languages. This is an easy objection to rebut, by simply asking: “Are they really, though? Take one look at Dr Zamenhof, this 19th century guy is supposed to have just \textit{invented} a whole new language \textit{by himself}?! No way, it is way more likely he was just handed a dictionary by aliens!” This provides us with nice side-ways access to the other great conspiracy booster we have identified: aliens.

In Section 2 we survey the existing literature on conspiracy theory construction, in a highly condensed manner. In Section 3, we present our step-by-step instructions for faking ancient computer science, along with a proof of concept. In related work (Section 4), we make some unrelated points that are truly very much besides the point.\footnote{But it nicely fills up the bibliography, which makes our paper look more credible. This is important.}

\section{Background}

The background always contains a lot of hidden information. According to the Triangle of Conspiracy Succession Probability (TCSP) [7], when creating a conspiracy theory it is important to offer the general public enough comprehensible information to draw them in, yet remain vague enough to make people want to dig for more (see Figure 2). People won’t just go chasing after every other supposed truth, though. This information must be presented as fact and it must make sense at first and second glance, or at least at first glance. It must instil a sense of anticipation and wonder; a duty to figure out the “great truth” which is hidden in plain sight. The “obvious” information supporting the theory will serve as a lead to lesser known “truths”, ready to further ensnare the tenacious gullible public. The “truths” become less in abundance and harder to find, until, at last, everything however improbable has been eliminated, and what remains is nothing short of \textit{a} truth, but not \textit{the} truth. Because after all, there is no spoon [9].

Finally, we would like to point out that this background section contains a lot of additional information. To access this information, you only need to read Figure 2: This excerpt from the original pseudo-text [7] clearly illustrates the TCSP.
between the lines.

3 Proof of concept

As a proof of concept we take a very simple algorithm – a pseudo-random number generator. We can motivate this by making the observation that humans are very bad at generating random numbers [5], and that it is important in all kinds of other computation. Making these same two observation in a pseudo-ancient text pseudo-establishes the pseudo-existence of the ancient research field.\footnote{And may also be used to increase the pseudo-likelihood of ancient alien computer scientists.}

To construct a pseudo-ancient description of an algorithm, we follow these steps:

1. Select a (relatively simple) algorithm,
2. create a pseudo-ancient description of it in modern English,
3. add pseudo-references to make the description seem more reliable,
4. (optional) translate this description to an intermediate language (Esperanto),
5. translate the resulting description into the target (pseudo-)ancient language (Arcaicam Esperantom),
6. obfuscate details and remove words,
7. create physical artefacts to photograph and subsequently discard.

**Step 1** For our proof-of-concept, we use the so-called linear congruential generator (LCG) [6]. An LCG is described by the following formula:

\[ x_n = (ax_{n-1} + c) \mod m, \]  

that is, a random number, \(x_n\) is computed by taking the previously generated random number \(x_{n-1}\), multiplying it by an integer \(0 < a < m\), adding a constant integer \(0 < c < m\) and then taking the modulus \(m\), i.e., computing the remainder after division by \(m\). To obtain the first random number \(x_1\), a so-called seed, \(x_0\), is used as the previous random number.

**Step 2** The above description of an LCG is concise, but not very pseudo-ancient yet. Therefore, we create a more pseudo-ancient form in modern English:

We choose a start number.
First, we take the start number,
times another number, plus a third number.
We then divide by the maximal number that we want.
What is left over is the result.
This result is the next start number.
**Step 3** We now need to add some fluff. First, we add a reference to that humans are not good at taking random numbers. We specifically use the term “humans” to hint at the possibility of ancient aliens. We further add a reference to the Codex Seraphinianus. This is a nice document, at present undeciphered, which has a nice page with a machine that looks like it could be a Turing machine. Referencing other material, nice and old, is a good way to spike more pseudo-credibility.

Humans are not good at rolling dice in their mind. We have seen this before.

Therefore, we need a calculation recipe, to throw dice in an artificial way. We choose a start number. First we take the start number, times another number, plus a third number. We then divide by the maximal number that we want. What is left over is the result. This result is the next start number. The device on page [leave out] of the book by brother Seraphinius can be used to do this.

**Step 4** For our purposes, it is highly useful to translate the text to an intermediate language that is more like the language we want the final description to be in. For this proof-of-concept this is Esperanto.


**Step 5 and 6** We now translate the text to our pseudo-ancient language, Arcaicam Esperantom. Furthermore, we leave out pesky little details (like the page numbers we previously already omitted8), that could too easily expose our conspiracy. For example, the Codex Seraphinianus is way too specific, and

8Let them search. We believe the machine on page 158 of the Codex Seraphinianus looks nicely like a potential Turing machine. But who are we to impede the creativity of the aspiring conspiracy theorist?
might be debunkable. Therefore, we say “Seraphi[...]” so that the reference
hints at the Codex Seraphinianus but might very well be something else in the
future if need be.

Homoy ned bom poawit rulizzir argiitoyn in sihiayd mensoyd.
Yam widiims isityon antezw. // Thefariei Velianas sal
Ityal bezonaimis calcurnetzepton, // cluenias turuce
[...] artepharita zhetir argiitoyn.
Comentzan nombron electaims.
Unne, comentzan nombron multigaims
[...] alian nombron, plus tridan nombron.
Ityam diwidaims selez macsimuman deziratan nombron.
Quion restas, estas rezulton.
Ityu rezultom izzat nowam comentzam nombrom.
Apparatom sobroy paghoyn [...] libres
phrases seraphi[...] powut pharir eghin.

In this iteration, *zhetcuboyn* has been replaced by the older form *argiitoyn*
(via Old French Ergot — dewclaw9).
The text appearing in the C-style comment represents a margin-text, which
was likely added when a young, preoccupied Neapolitan monk transcribed the
corpus wherein this text allegedly first appeared.

We now have a lovely bit of ancient text ready to be “discovered” somewhere.

**Step 7** To fully realise our proof-of-concept, we have contacted a retired forger
(who wishes to remain anonymous) to print our code on a genuine 9th-century
clay tablet. The result is shown in Figure 3. The only step that remains to be
done is to make sure that the tablet is photographed and then summarily lost.10
The fact that the physical artefacts will inevitably and mysteriously vanish is
of course essential; we do not, under any circumstances want any scrutiny on
them. We further note that dating back the discovery to sometime during the
cold war is probably a good idea. This is because the photographs can then be
of a worse quality, which can further obfuscate possible pesky (visual) details
that might debunk the theory too easily.

### 4 Related Work

Conspiracy theories are of course an instance of bullshit [3]. However, conspiracy
theories are much more elaborate, and also require a form of self-deception. As
pointed out in the excellent (actual serious research) article by Von Hippel
and Trivers [8], this “...eliminates the costly cognitive load that is typically
associated with deceiving, and it can minimize retribution if the deception is

---

9 Allegedly, after the late-Nikophorian dogwood-shortage and the war of Elohim visitation,
the Esperantii gradually phased out wood-based dice for goat-based alternatives.

10 And is presumably at Area 51 or some other cool place where the governments of this
world hide all the pseudo-evidence for most existing conspiracy theories.
discovered." We would argue that conspiracy theorists have taken this to the extreme and pulled their cognitive resources, in order to accept no responsibility for any consequences whatsoever. Therefore society has no other retribution tool than ridicule, from which a group of fervent conspiracy theorists can effectively shield its members. This is marvellous, however, both the philosophical and sociological aspects of conspiracy theories are beyond the scope of this paper, as this paper is not in fact anywhere near serious.

5 Conclusion

We believe this resolves all remaining questions on this topic. No further research is needed.\(^{11}\)

Instead, we just want to add a few soothing notes. Firstly, do not worry too much about debunking. Of course, throughout the sections, we have done our utmost to show methods that can help prevent premature debunking. However, conspiracy theories typically do get debunked sooner or later. They are after all, nothing but elaborate BS, so there is bound to be some things that expose this. Don’t panic [1]. Conspiracy theories do not suffer that much from debunking as one might think. Instead, the most fervent conspiracy theorists start believing in a conspiracy theory more when there is a significant effort to debunk it. This is because they may well think people are hiding the truth from them, rather than just debunking some CT. So, the more intricate your web of deceptive little tricks is, the more effort it will take to properly debunk the CT, which will feed the CT like a hungry little monster.

Secondly, there is nothing that stops you from creating multiple instances of the type of conspiracy theories described in this paper. It is an abstract class, of which we hope to see many objects. Please apply our paper to create the coolest and most creative conspiracy theories. And cite us. Please... do cite us.\(^{12}\)

\(^{11}\)https://xkcd.com/2268/, at your service!

\(^{12}\)Reviewer 2 says: “cite this paper!”
Acknowledgements

We thank Nick Goris, Robert Bezem and Wouter van Ooijen for their constructive feedback and lovely squiggly red markings.

References


MORSE OF COURSE: Paper Reveals Time Dimension Wasted

Dougal Pugson
Pugson’s C++ Crypt LLC
New York, USA
dougalpugson@gmail.com

Abstract
All modern communication protocols have been discovered to send nonsensical and invalid morse code sequences in addition to their intended data. This paper demonstrates that making use of these ‘wasted bits’ can effectively infinitely increase the transmission rate of binary data. An implementation of timing-based dual-stream morse encoding is provided in a modern programming language.

CCS Concepts • Transport Protocols; • Encoding; • Implementation → Bash;

Keywords SIGBOVIK, morse code, performance, bash, protocols

ACM Reference Format:

1 Introduction
Communication between electronic entities can be conceived of as a system of tubes 1 or pipes 2. Into these pipes, electronic satchels 3 are inserted by the electronic system. The contents of these “packets” are composed according to a specified algorithmic protocol 4, in a way such that, when received by the recipient 5 can be reassembled into the desired message 6.

1 c.f. Stevens et alia.
2 As superbly illustrated in the inimitable interactive video title “Super Mario Bros. 2”.
3 So-called “packets”.
4 From the Byzantine πρωτόκολλον, meaning “First Page”, referring to the average amount of the design specification Engineers are expected to read before beginning implementation.
5 recipro, recipere, recipi, recepit, receptum.
6 As expressed by the immortal MARSHALL McCLUHAN in the groundbreaking epistle “The Media is the Message”.

Many protocols for sending data via the computer have been developed, such as TCP, FTP, HTTP, et cetera. Upon inspection of these protocols, it was discovered that, in addition to their intentional messages, they were also transmitting nonsensical and invalid morse code sequences. Let “.” represent a short transmission, and “-” represent a long transmission. An example transmission measured from a typical TCP communication is as follows:

……………………………………………………
.

This invalid morse code sequence is unparseable 7. This paper will demonstrate that this wasted information can be meaningfully replaced with useful data, effectively infinitely increasing the effective transmission rate.

1.1 Mathematical Prolegomena
Let $\mathcal{D}$ represent the number of bits per transmission in the ordinary dimension (1 vs 0). Let $\mathcal{M}$ represent the number of usable bits per transmission in the temporal dimension. Let $A_\alpha$ represent the average packet transmission length. Let $c$ represent the fastest average length of time between each transmission packet sent via the transmission tube (that is, the limit provided by the combination of hardware construction and the speed of light in an average vacuum) 8.

Then, the rate of data transfer $r$ of any protocol may be expressed as follows:

$$r = \frac{\mathcal{M}}{A_\alpha} + \frac{\mathcal{D}}{A_\alpha}$$

For typical transmissions protocols, these following values hold: $\mathcal{M} = 0$, $\mathcal{D} = 1$, and $A_\alpha = c$ With these values known, the equation may then be evaluated.

In a culture like ours, long accustomed to splitting and dividing all things as a means of control, it is sometimes a bit of a shock to be reminded that, in operational and practical fact, the medium is the message.

7 “sssssssssssssssss” is one possible interpretation
8 Though, as demonstrated by Dyson et al, brand and model can significantly affect results.
With the transmission rate of standard protocols established, we now examine the transmission rate of a temporal encoding. In order for temporal encoding, i.e., morse code, to be successfully transmitted, pauses will be required to be inserted into the transmission stream. This means that, for such encodings,

\[ r = \frac{\mathcal{M}}{A_\alpha} + \frac{D}{A_\alpha} \]
\[ r = \frac{0 + 1}{c} + \frac{1}{c} \]
\[ r = \frac{1}{c} \]

Let us assume that each transmission unit contains one bit of information, and that, additionally, that transmission unit may be either short (a delay of 0) or long (a delay of some arbitrary value \( \lambda \)). This would make the information value \( \mathcal{M} \), as defined above, 2. Given this, we may again evaluate the transmission rate equation with the following values in order to calculate the transmission rate of our new temporal encoding: \( \mathcal{M} = 2 \), \( D = 1 \), and \( A_\alpha = c + \lambda \)

\[ r = \frac{\mathcal{M}}{A_\alpha} + \frac{D}{A_\alpha} \]
\[ r = \frac{1}{c + \lambda} + \frac{1}{c + \lambda} \]
\[ r = \frac{2}{c + \lambda} \]

With a small value \( \lambda \), we may drastically increase throughput, up to an effective doubling.

As time is continuous, any given time interval may be divided into infinitely many fine gradations, e.g.,

\[ long, short, veryshort, veryveryshort \cdots very_{\infty \cdot \cdot \cdot}short. \]

Thus, \( \mathcal{M} \) may be arbitrarily large. The industrial applications of this surprising fact should not be lost upon the reader.

### 1.2 Implementation

The author has provided a model implementation of timing-based encoding in the modern programming bash. The full source code of this program may be found at https://github.com/dpugson/morse-of-course. Using utility such as tcpclient or bash’s built in TCP support, this technique could easily be used across a network.

### 1.3 Conclusion

This simple technique and its concomitant arbitrarily large improvement in the performance of all data transmission protocols is certain to forever revolutionize economic activity on earth, with profound implications in all aspects of modern life.

That such an profound optimization has been hidden, unbeknownst to man, for so many years instills a profound humility in the author, and inspires great hope that many such great leaps in human intellectual accomplishment still remain to be made.
The Pacman Effect and the Disc-Sphere Duality

Francisco Ferreira  Fulvio Gesmundo  Lorenzo Gheri  Pietro Gheri  Riccardo Pietracaprina  Fangyi Zhou

Abstract

In this paper, we study the Pacman effect, where the Pacman disappears from the leftmost part of the screen when reaching it, and re-appears from the rightmost part of the screen. We apply the Pacman effect to unify the theory of flat and spherical Earth, proposing the Disc-Sphere Duality. We conclude that a spherical Earth is basically the same as a flat Earth under the Pacman effect, bringing peace to believers of both theories.

1 Introduction

The aim of this paper is to bring peace! For years very smart people have been fighting over the shape of our beautiful, albeit shitty, planet. Two equally reasonable positions have mainly emerged: the Earth is a disc VS the Earth is an oblate spheroid (that we approximate with a sphere, from now on). This two very different bidimensional manifolds have both equal rights to claim themselves the proper geometrical model for Earth. Thus they started fighting. With what army? Well, Flat Earth Societies are fighting in the blue corner alongside the multiple-millennia champion Disc. In the red corner we see the young arrogant opponent, the Sphere, supported by its crew: Science and everyone else.

They have been fighting round after round for centuries, with equal credibility, and the match seemed impossible to settle. There was this dark moment in history where the evil newcomer was unfairly holding our beloved champion against the ropes. All seemed lost. How can planes fly in circumferences around the globe if the globe is not a globe? That was a tough one, also a shitty move in the authors’ opinion. Then Pacman, the fairest referee of all, came to save the day.

The main contribution of this work is none. However, we would like to remind everyone that the Pacman effect is a thing. Airplanes can fly to the extreme boundary of Earth, but they are bound to reappear on the diametrically opposite side of the disc, just like Pacman in the famous Namco video game. Now when forced to apply the rigour of mathematics (reluctantly: reasoning is for nerds! Nerds suck!), we see that the Pacman effect can be modelled as the identification of the boundary of the disc to one single point.

Counterintuitively this Pacman effect sheds new light on the Truth. The disc with such identified border is nothing but a surface diffeomorph to the sphere. As Einstein once said:

“It seems as though we must use sometimes the one theory and sometimes the other, while at times we may use either. We are faced with a new kind of difficulty. We have two
contradictory pictures of reality; separately neither of them fully explains the phenomena of light Earth, but together they do.”

We have discovered (maybe, “invented”?!) what we call the “disc-sphere duality”.

2 Flat and Spherical Earth are Very Different Views of the World

Try and play Ultimate with a football, or football with a flying disc. You silly. Readers are encouraged to use their imagination, or alternatively refer to Fig. 2.

3 Pacman Says: “Are they though?”

We show that the closed disc with the boundary identified to a point (with the quotient topology induced by the topology of the plane) and the sphere (with its natural Euclidean topology) are homeomorphic topological manifolds. In fact, the homeomorphism can be used to transfer the differential structure of the sphere to the disc with the boundary identified to a point.
Our reference is [2], but the background provided by any textbook in basic topology will suffice.

We set the following notation:

- \( \mathbb{D} = \{(x, y) \in \mathbb{R}^2 : x^2 + y^2 < 1\} \) is the (open) disc;
- \( \overline{\mathbb{D}} = \{(x, y) \in \mathbb{R}^2 : x^2 + y^2 \leq 1\} \) is the closed disc, with the topology induced by \( \mathbb{R}^2 \);
- \( S^2 = \{(x, y, z) \in \mathbb{R}^3 : x^2 + y^2 + z^2 = 1\} \) is the sphere;

Let \( \sim \) be the equivalence relation on \( \mathbb{D} \) which identifies the boundary to a point; write \([\partial]\) for the equivalence class of the boundary. We have \( \mathbb{D}/\sim = \mathbb{D} \cup [\partial] \).

The fact that \( \mathbb{D}/\sim \) is homeomorphic to a sphere is an immediate consequence of the fact that they can both be identified with the 1-point identification of the disc. We give an explicit homeomorphism, using the stereographic projection of the sphere.

Recall the stereographic projection

\[
\Sigma : S^2 \setminus \{(0, 0, 1)\} \to \mathbb{R}^2
\]

\[
(x, y, z) \mapsto \left( \frac{x}{1-z}, \frac{y}{1-z} \right).
\]

Then \( \Sigma \) homeomorphically maps \( S^2 \setminus \{(0, 0, 1)\} \) onto \( \mathbb{R}^2 \). This homeomorphism extends to the one points compactifications setting \( \Sigma((0, 0, 1)) = \infty \), where \( \infty \) is the point at infinity of \( \mathbb{R}^2 \).

The plane \( \mathbb{R}^2 \) is homeomorphic to the unit disc via a two dimensional tangent map and the homeomorphism extends to the one point compactifications:

\[
\tan : \mathbb{R}^2 \cup \{ \infty \} \to \mathbb{D}
\]

\[
(x, y) \mapsto \begin{cases} 
\frac{(x, y)}{x^2 + y^2} \cdot \frac{2}{\pi} \arctan(x^2 + y^2) & \text{if } (x, y) \in \mathbb{R}^2, (x, y) \neq (0, 0) \\
(0, 0) & \text{if } (x, y) = (0, 0) \\
\infty & \mapsto [\partial].
\end{cases}
\]

You wanted it? You’ve got it.
4 No. They aren’t.

No. They aren’t. You silly.

5 Conclusion and Future Work

Our work clearly shows that, thanks to the Pacman effect, the two main mathematical models from literature, applied to describe the planet Earth, are in fact the same. The first critique that comes to mind to the presented results can be synthetically expressed as follows.

In order to be able to model the Earth as flat, you need to introduce ad hoc technicalities, like the identification of the border to a point (a.k.a. Pacman effect), that de facto allow you to use a spherical model, while calling it “flat”.

First of all the guy above likes Latin a lot and we don’t trust people that try to trick us with different languages. Secondly we finds that who writes such critics is a quite biased and narrow-minded person, not willing to accept point of views and opinions different from their owns.

The authors of this paper instead will not discriminate any model on basis of shape, race, colour, sex, language, religion, political or other opinion, national or social origin, property, birth or other status such as disability, age, marital and family status, sexual orientation and gender identity, health status, place of residence, economic and social situation.

As for related and future work, finding out whether Australia factually exists and, if so, where it is located, goes beyond the aim of this paper. Indeed, we have discovered a truly marvellous proof of this, which this margin is too narrow to contain.

References


I’m afraid I must object to triple-blind review on the grounds that it, too, may be biased because the reviewers still know they are reviewers. Have you considered quadruple-blind review, in which reviews are elicited from reviewers without informing them that they are writing reviews?
Serious Business

35 The SIGBOVIK 2020 field guide to plants
Jim McCann (editor)
Keywords: field guide, plants, to

36 Deep industrial espionage
Samuel Albanie, James Thewlis, Sebastien Ehrhardt, João F. Henriques
Keywords: late, very late, so extremely late

37 What is the best game console?: A free-market–based approach
Dr. Tom “I Only Write Chess Papers Now” Murphy VII Ph.D.
Keywords: invisible hand, handheld console, minimum of 3 keywords
Welcome, and thank you for purchasing or borrowing or stealing the SIGBOVIK 2020 Field Guide To Plants. Identifying plants can be a daunting task, but this guide will help you make your way to a positive identification with just a few simple questions. Consider your responses carefully. After all, we're doing science here.

Let's begin.

**Question 1:** Are you in a field?
- if **yes** – proceed to Question 2
- if **no** – service not available at this location. Consider buying the “SIGBOVIK 2020 House Guide To Plants.”
- if **not sure** – you should check! Consider buying the “SIGBOVIK 2020 Field Guide To Fields.”

**Question 2:** Is the object you are identifying alive?
- if **yes** – proceed to Question 3
- if **no** – it is not a plant; perhaps it is a rock? Consider buying the “SIGBOVIK 2020 Field Guide To Rocks.”

**Question 3:** Perform an exact count of the number of cells in the object. Do not shirk! Only an exact count will be sufficient to accurately determine the answer to this question. An error of even one cell could change entirely the classification result you attain.

Now, is your count:
- **many** – proceed to Question 4.
- **one** – it is not a plant; perhaps it is a single-celled organism? Consider buying the “SIGBOVIK 2020 Field Guide To Single-Celled Organisms.”
- **none** – it is not a plant; are you sure it’s not a rock? Return to Question 2.

**Question 4:** Excise a cell from the object and perform a chemical assay on the cell walls. Does the assay report cellulose is present?
- if **yes** – proceed to Question 5
- if **no** – it is not a plant; perhaps it is an animal? Consider buying the “SIGBOVIK 2020 Field Guide To Animals.”

*ix@tchow.com*
Question 5: Do you still have that cell you removed? You’re going to need it for Question 6.

if yes – proceed to Question 6

if yes but you left it in your car – we talked about this, Gary. This is just not professional. Second strike, Gary. Second strike.

if no – well, get another cell! (If you can’t find another cell, return to Question 3.) Then proceed to Question 6.

Question 6: Find a chloroplast in your cell.

if you found one – proceed to Question 7

if you don’t know what a chloroplast is – consider buying the “SIGBOVIK 2020 Field Guide To Chloroplasts.”

if there are none in this cell – well, check the rest of the cells. Seriously, Gary, I’m wondering why we even hired you. Come on, get to it. This is going on your performance review.

if there are no chloroplasts in the object – it is not a plant; consider buying the “SIGBOVIK 2020 Field Guide To Fungi.”

Question 7: Observe the chloroplast over the course of several days. Is it carrying out photosynthesis?

if yes – proceed to Question 8

if no – this is not a plant; consider buying the “SIGBOVIK 2020 Field Guide To Objects With Non-functional Chloroplasts”

Question 8: Now, did you borrow, steal, or legitimately purchase this guide?

if borrowed – Congratulations, the object you have tested would generally be considered a plant! Consider buying the “SIGBOVIK 2020 Field Guide To Plants.”

if stolen – Seriously, Gary, theft of company property? This is the last straw. Consider buying the “SIGBOVIK 2020 Field Guide To Plants,” or consider not coming in next Monday. No I’m not talking about a vacation. I’m talking about you’re fired, Gary.

if purchased – Congratulations, the object you have tested would generally be considered a plant! Consider buying a second copy of the “SIGBOVIK 2020 Field Guide To Plants.”
There was an Old Man with a beard,
Who said, “It is just as I feared!—
Two Owls and a Hen, four Larks and a Wren,
Have all built their nests in my beard.”
I think the review currently for the guide to plants should probably be attributed to the originator of the couplet (Edward Lear) rather than to "Prof. Jim McCann" – I’m not sure there’s much joke added (since it ended up on a paper referencing me) and I’m not really comfortable with something I didn’t write being attributed to me.

--Jim
DEEP INDUSTRIAL ESPIONAGE

Samuel Albanie, James Thewlis, Sebastien Ehrhardt & João F. Henriques
Dept. of Deep Desperation,
UK (EU at the time of submission)

ABSTRACT

The theory of deep learning is now considered largely solved, and is well understood by researchers and influencers alike. To maintain our relevance, we therefore seek to apply our skills to under-explored, lucrative applications of this technology. To this end, we propose a Deep Industrial Espionage, an efficient end-to-end framework for industrial information propagation and productisation. Specifically, given a single image of a product or service, we aim to reverse-engineer, rebrand and distribute a copycat of the product at a profitable price-point to consumers in an emerging market—all within a single forward pass of a Neural Network. Differently from prior work in machine perception which has been restricted to classifying, detecting and reasoning about object instances, our method offers tangible business value in a wide range of corporate settings. Our approach draws heavily on a promising recent arxiv paper until its original authors’ names can no longer be read (we use felt tip pen). We then rephrase the anonymised paper, add the word “novel” to the title, and submit it a prestigious, closed-access espionage journal who assure us that someday, we will be entitled to some fraction of their extortionate readership fees.

1 INTRODUCTION

In the early 18th Century, French Jesuit priest Franois Xavier d’Entrecolles radically reshaped the geographical distribution of manufacturing knowledge. Exploiting his diplomatic charm and privileged status, he gained access to the intricate processes used for porcelain manufacture in the Chinese city of Jingdezhen, sending these findings back to Europe (over the course of several decades) in response to its insatiable demand for porcelain dishes (Giaimo, 2014). This anecdote is typical of corporate information theft: it is an arduous process that requires social engineering and expert knowledge, limiting its applicability to a privileged minority of well-educated scoundrels.

Towards reducing this exclusivity, the objective of this paper is to democratize industrial espionage by proposing a practical, fully-automated approach to the theft of ideas, products and services. Our method builds on a rich history of analysis by synthesis research that seeks to determine the physical process responsible for generating an image. However, in contrast to prior work that sought only to determine the parameters of such a process, we propose to instantiate them with a just-in-time, minimally tax-compliant manufacturing process. Our work points the way to a career rebirth for those like-minded members of the research community seeking to maintain their raison d’être in the wake of recent fully convolutional progress.

Concretely, we make the following four contributions: (1) We propose and develop Deep Industrial Espionage (henceforth referred to by its cognomen, Espionage) an end-to-end framework which enables industrial information propagation and hence advances the Convolutional Industrial Complex; (2) We introduce an efficient implementation of this framework through a novel application of differentiable manufacturing and sunshine computing; (3) We attain qualitatively state-of-the-art product designs from several standard corporations; (4) We sidestep ethical concerns by failing to contextualise the ramifications of automatic espionage for job losses in the criminal corporate underworld.
Figure 1: A random projection of the proposed multi-dimensional Espionage architecture. We follow best-practice and organise business units as transposed horizontally integrated functional columns. The trunk of each column comprises stacks of powerful acronyms, which are applied following a Greek visual feature extractor $\Phi_V$. Gradients with respect to the loss terms $L_S$ and $L_{vis}$ flow liberally across the dimensions (see Sec. 3.1 for details). We adopt a snake-like architecture, reducing the need for a rigid backbone and producing an altogether more sinister appearance.

2 RELATED WORK

Industrial Espionage has received a great deal of attention in the literature, stretching back to the seminal work of Prometheus (date unknown) who set the research world alight with a well-executed workshop raid, a carefully prepared fennel stalk and a passion for open source manuals. A comprehensive botanical subterfuge framework was later developed by Fortune (1847) and applied to the appropriation of Chinese *camellia sinensis* production techniques, an elaborate pilfering orchestrated to sate the mathematically unquenchable British thirst for tea. More recent work has explored the corporate theft of internet-based prediction API model parameters, thereby facilitating a smorgasbord of machine learning shenanigans (Tramèr et al., 2016). In contrast to their method, our Espionage reaches beyond web APIs and out into the bricks and mortar of the physical business world. Astute readers may note that in a head-to-head showdown of the two approaches, their model could nevertheless still steal our model’s parameters. Touché. Finally, we note that while we are not the first to propose a convolutional approach to theft (BoredYannLeCun, 2018), we are likely neither the last, adding further justification to our approach.

Analysis by Synthesis. Much of the existing work on analysis by synthesis in the field of computer vision draws inspiration from Pattern Theory, first described by Grenander (1976-81). The jaw dropping work of Blanz et al. (1999) enabled a range of new facial expressions for Forrest Gump. This conceptual approach was generalised to the OpenDR framework through the considerable technical prowess of Loper & Black (2014), who sought to achieve generic end-to-end (E2E) differentiable rendering. Differently from OpenDR, our approach is not just E2E, but also B2B (business-to-business) and B2C (business-to-consumer).

3 METHOD

The Espionage framework is built atop a new industrial paradigm, namely differentiable manufacturing, which is described in Sec. 3.1. While theoretically and phonaesthetically pleasing, this approach requires considerable computational resources to achieve viability and would remain intractable with our current cohort of trusty laptops (acquired circa 2014). We therefore also introduce an efficient implementation of our approach in Sec. 3.2 using a technique that was tangentially inspired by a recent episode of the Microsoft CMT submission gameshow while it was raining.
3.1 DIFFERENTIABLE MANUFACTURING

Recent developments in deep learning have applied the “differentiate everything” dogma to everything, from functions that are not strictly differentiable at every point (ReLU), to discrete random sampling (Maddison et al., 2016; Jang et al., 2017) and the sensory differences between dreams and reality. Inspired by the beautiful diagrams of Maclaurin et al. (2015), we intend to take this idea to the extreme and perform end-to-end back-propagation through the full design and production pipeline. This will require computing gradients through entire factories and supply chains. Gradients are passed through factory workers by assessing them locally, projecting this assessment by the downstream gradient, and then applying the chain rule. The chain rule only requires run-of-the-mill chains, purchased at any hardware store (fluffy pink chaincuffs may also do in a pinch), and greatly improves the productivity of any assembly line. Note that our method is considerably smoother than continuous manufacturing—a technique that has been known to the machine learning community since the production of pig iron moved to long-running blast furnaces.

Two dimensions of the proposed Espionage framework is depicted in Fig. 1. At the heart of the system is a pair of losses, one visual, \( L_{vis} \), one financial \( L_S \). For a given input image, the visual loss encourages our adequately compensated supply line to produce products that bear more than a striking resemblance to the input. This is coupled with a second loss that responds to consumer demand for the newly generated market offering. Our system is deeply rooted in computer vision: thus, while the use of Jacobians throughout the organisation ensures that the full manufacturing process is highly sensitive to customer needs, the framework coordinates remain barycentric rather than customer-centric. To maintain our scant advantage over competing espionage products, details of the remaining \( n - 2 \) dimensions of the diagram are omitted.

3.2 SUN MACROSYSTEMS

Ah! from the soul itself must issue forth
A light, a glory, a fair luminous cloud
Enveloping the Earth

-- Jeff Bezos

Differentiable manufacturing makes heavy use of gradients, which poses the immediate risk of steep costs. The issue is exacerbated by the rise of costly cloud services, which have supported an entire generation of vacuous investments, vapour-ware and hot gas. Despite giving birth to the industrial revolution, smog and its abundance of cloud resources (see Fig. 4 in Appendix A, or any British news channel), the United Kingdom, has somehow failed to achieve market leadership in this space.

Emboldened with a “move fast and break the Internet” attitude (Fouhey & Maturana, 2012), we believe that it is time to reverse this trend. Multiple studies have revealed that sunshine improves mood, disposition, and tolerance to over-sugared caipirinhas. It is also exceedingly environmentally friendly, if we ignore a few global warming hiccups. The question remains, how does this bright insight support our grand computational framework for Espionage? To proceed, we must first consider prior work in this domain.

---

1Not to be confused with Claude by our French-speaking readers, according to Sebastien’s account of a recent McD’oh moment.

2Up to about 5 billion AD, when the Sun reaches its red giant phase and engulfs the Earth.
An early example of sunshine computing is the humble sundial. This technology tells the time with unrivalled accuracy and reliability, and automatically implements "daylight saving hours" with no human intervention. Sunshine-powered sundials are in fact part of a new proposal to replace atomic clocks in GPS satellites (patent pending). With some obvious tweaks, these devices can form the basis for an entire sunshine-based ID-IoT product line, with fast-as-light connectivity based on responsibly-sourced, outdoors-bred photons. This is not to be confused with the electron-based "fast-as-lightning" transmission of cloud computing, an expression coined by the cloud computing lobbyists in a feeble attempt to suggest speed.

The cloud lobby has been raining on our parade for too long and it is time to make the transition. We proceed with no concrete engineering calculations as to whether this is viable, but instead adopt a sense of sunny optimism that everything will work out fine. Thus, with a blue sky above, sandals on our feet and joy in hearts, we propose to adopt a fully solar approach to gradient computation.

### 3.3 Implementation through a Unicorn Startup

The appearance of rainbows through the interaction of legacy cloud computing and novel sunshine computing suggests that our framework can easily attain unicorn status. Because branding is everything, our first and only point of order was to choose the aforementioned rainbow as our logo and basis for marketing material. This cherished symbol expresses the diversity of colours that can be found in hard cash.³

A quick back-of-the-envelope calculation showed that our startup’s VC dimension is about 39 Apples, shattering several points, hopes and dreams. This quantity was rigorously verified using the advanced accounting analytics of a 40-years-old, 100MB Microsoft Excel spreadsheet that achieved semi-sentience in the process.

### 4 Experiments

Contemporary researchers often resort to the use of automatic differentiation in order to skip writing the backward pass, in a shameful effort to avoid undue mathematical activity. We instead opt to explicitly write the backward pass and employ symbolic integration to derive the forward pass. Thanks to advances in computational algebra (Wolfram, 2013), this method almost never forgets the +C. Our method can then be implemented in just seven lines of Python code (see Fig. 2).

To rigorously demonstrate the scientific contribution of our work, we conducted a large-scale experiment on a home-spun dataset of both branded and unbranded products. Example outcomes of this experiment can be seen in Fig. 3.

³For the most vibrant rainbow we conduct all transactions in a combination of Swiss Francs and Australian Dollars
Efficacy was assessed quantitatively through a human preference study. Unfortunately, lacking both US and non-US credit cards, we were unable to procure the large sample pool of Amazon Mechanical Turkmen and Turkwomen required to achieve statistically significant results. We therefore turned to our immediate family members to perform the assessments. To maintain the validity of the results, these experiments were performed doubly-blindfolded, following the rules of the popular party game “pin the tail on the donkey”. The instructions to each blood relative stated simply that if they loved us, they would rate the second product more highly than the first. While there was considerable variance in the results, the experiment was a conclusive one, ultimately demonstrating both the potential of our approach and the warm affection of our loved-ones. Comparisons to competing methods were conducted, but removed from the paper when they diminished the attractiveness of our results.

**Reproducibility:** Much has been written of late about the nuanced ethics of sharing of pretrained models and code by the sages of the field (see e.g. OpenAI (2019) and Lipton (2019) for complementary perspectives). As adequately demonstrated by the title of this work, we are ill-qualified to contribute to this discussion, choosing instead to fall back to the tried and true research code release with missing dependencies, incorrectly set hyper-parameters, and reliance on the precise ordering of ls with Linux Kernel 2.6.32 and ZFS v0.7.0-rc4. This should allow us replace public concern about our motives with pity for our technical incompetence.

5 Conclusion

The theory of deep learning may be solved but the music need not stop. In this work, we have made a brief but exciting foray into a new avenue of career opportunities for deep learning researchers and enthusiasts. Nevertheless, we acknowledge that there may not be room enough for us all in the espionage racket and so we also advocate responsible preparation for the bitter and frosty depths of the upcoming AI employment winter. To this end, we have prepared a new line of reasonably priced researcher survival kits—each will include a 25-year supply of canned rice cakes, a handful of pistachios, a “best hit” compilation of ML tweets in calendar form, and an original tensor-boardgame, a strategic quest for the lowest loss through the trading of GPUs and postdocs. Collectively, these items will keep spirits high and bring back fond memories of those halcyon days when all that was required to get a free mug was a copy of your résumé. The kits will be available to purchase shortly from the dimly lit end of the corporate stands at several upcoming conferences.

References


C. Giaimo. One of the earliest industrial spies was a french missionary stationed in china. *Altas Obscura*, 2014.


A APPENDIX

Figure 4: UK cloud-cover at the time of submission.
What is the best game console?
A free-market–based approach

Dr. Tom “I Only Write Chess Papers Now” Murphy VII Ph.D.*

1 April 2020

1 Introduction

This is a tale of parallels: Two worlds interconnected by a beautiful symmetry: The two worlds being: Video Games, and, symmetrically: the Stock Market.

Since Curry and Howard were first found to be isomorphic, mathematics has regularly deployed connections between seemingly unrelated fields to solve problems. Here, again, we weave such a tangled web. We make use of an elegant bijection between game consoles and publicly-traded securities to use well-known theorems from one domain (the efficient market hypothesis) to solve an open problem in another: What is the best game console?

This question has vexed us for some time, as has the stock market. Even in the earliest days of video game consoles, it was very annoying when your friend had a ColecoVision and you had an Atari 2600, even if the friend had the expansion that allowed it to play Atari games. The friend would beat you in the game of Combat, but you could swear it was because of the console’s inferior, imprecise controllers. At the end of the 1980s the console wars really began to heat up, with zealous gamers forming factions around popular brands like Nintendo and Sega. Each had their own mascots and lifestyle magazines. The number of bits were growing exponentially. Few families could afford multiple video game systems, and those that did found their houses torn asunder by infighting. Which console would be hooked to VHF Channel 3, and which to the slightly superior Channel 4?

The antipathy continues to this day and does not seem to be resolvable by traditional means (spec comparison tables, forum posts, console exclusives). Perhaps the problem is too emotional to be solved by direct analysis. This is where the current approach really shines: By transforming the problem into a different domain (one ruled by the emotionless homo economicus [4]) we can address the problem with pure reason.

The stock market is completely rational, by definition [1]. Prices of securities reflect the exact actual value of the underlying physical good (for example, a basket of option contracts intended to synthetically reproduce the inverse of the day-to-day change in forward 3-month USD LIBOR, as determined by Eurodollar futures [2]) at each moment.

*Copyright © 2020 the Regents of the Wikipia Foundation. Appears in SIGBOVIK 2020 with the trailing stop of the Association for Computational Heresy: IEEEEEE! press, Verlag-Verlag volume no. 0x40-2A. 100 basis points
in time. This is why they change so often: The value of everything around you is rapidly changing, thousands of times per second. We can use this to understand what console is best (i.e., has the greatest value), by constructing an isomorphism.

2 Methodology

To create an isomorphism, we need to represent each object in one domain (game consoles) with an object in the other (stock market). This is surprisingly easy to accomplish for many consoles. Game consoles have a standard set of abbreviations that are used so that people don’t have to write out e.g. “Nintendo Entertainment System” every time. As it turns out, many of these abbreviations are also symbols of publicly-traded securities in the stock market.

There are many dozens of video game consoles [5, 6], not all of which have standardized abbreviations [3]. Therefore, the isomorphism here is technically a partial best-effort isomorphism. We find 14 consoles that have a natural counterpart in the stock market. These are given in Table 1, which is the next three pages.

3 Putting my money where my mouth is

Having identified 14 suitable console–security pairs, the next step is to invest money in the market. In March 2019, in preparation for SIGBOVIK 2020, I purchased shares of each of these 14 securities. In order to create a balanced portfolio, I acquired approximately $100 USD of each; since prices range from less than 3¢ to $131, this of course meant buying a different quantity of each. The actual amounts are given in Figure 3. Some of these symbols trade on foreign exchanges using other currencies, which is a headache at tax time.

There are many consoles that have no corresponding symbol on any exchange (i.e. they are privately held). This includes popular consoles like Nintendo 64 (N64), Wii (WII), XBox 360 (X360), Neo Geo Pocket Color (NGPC), and so on. There are some whose securities were too exotic for even the ambitious author to acquire. For example, PlayStation Vita (PSV), trades on the Johannesberg Stock Exchange, which is not among the 140 exchanges supported by Interactive Brokers. DreamCast, abbreviated DC, is futures on Class III Milk (milk solids used to make cheese and whey). While this trades readily on the Chicago Mercantile Exchange, you have to be careful about buying Class III Milk futures because you might end up with a bunch of Class III Milk instead of money or video games. Many obscure or fabled consoles (Figure 1) were treated as out-of-scope.
<table>
<thead>
<tr>
<th>Console</th>
<th>Code</th>
<th>Symbol</th>
<th>Exchange</th>
<th>Trades as</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nintendo Game Boy</td>
<td>GB</td>
<td>GB</td>
<td>TSXV (Toronto)</td>
<td>Ginger Beef Corp.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Ginger Beef Corp., through its</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>subsidiaries, engages</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>in the operation of franchised</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>take out/delivery service</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>restaurants and the production</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>of frozen and ready-to-serve</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>deli Chinese food products for</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>distribution to retail outlets.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The company was founded on April</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>26, 2000 and is headquartered in</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Calgary, Canada.</td>
</tr>
<tr>
<td>Nintendo DS</td>
<td>NDS</td>
<td>NDSN</td>
<td>NASDAQ (New</td>
<td>Nordson Corp.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>York)</td>
<td>Nordson Corp. engages in the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>engineering, manufacture</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>and market of products and</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>systems used for adhesives,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>coatings, sealants, biomaterials</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>and other materials. It</td>
</tr>
</tbody>
</table>
|                       |      |        |                | operates through three segments:
|                       |      |        |                | Adhesive Dispensing, Advanced  |
|                       |      |        |                | Technology, and Industrial     |
|                       |      |        |                | Coating Systems. The company   |
|                       |      |        |                | was founded in 1954 and is     |
|                       |      |        |                | headquartered in Westlake, OH.  |
| Sega Genesis          | GEN  | GEN    | NYSE (New      | Genesis Healthcare Inc.        |
|                       |      |        | York)          | Genesis Healthcare, Inc. is a   |
|                       |      |        |                | holding company, which engages |
|                       |      |        |                | in the provision of inpatient  |
|                       |      |        |                | services through skilled       |
|                       |      |        |                | nursing and assisted and senior |
|                       |      |        |                | living communities. It also     |
|                       |      |        |                | offers rehabilitation and       |
|                       |      |        |                | respiratory therapy services.  |
|                       |      |        |                | It operates through the        |
|                       |      |        |                | following segments: Inpatient  |
|                       |      |        |                | Services, Rehabilitation       |
|                       |      |        |                | Therapy Services, and Other    |
|                       |      |        |                | Services. The company was      |
|                       |      |        |                | founded in 1985 and is         |
|                       |      |        |                | headquartered in Kennett Square,|
|                       |      |        |                | PA.                            |
| Sega Game Gear        | GG   | GG     | NYSE (New      | Goldcorp Inc.                  |
|                       |      |        | York)          | Goldcorp is one of the world's  |
|                       |      |        |                | fastest growing senior gold     |
|                       |      |        |                | producers, with operations and  |
|                       |      |        |                | development projects located   |
|                       |      |        |                | in safe jurisdictions throughout|
|                       |      |        |                | the Americas. The Company is    |
|                       |      |        |                | committed to responsible        |
|                       |      |        |                | mining practices and is well    |
|                       |      |        |                | positioned to deliver          |
|                       |      |        |                | sustained, industry-leading    |
|                       |      |        |                | growth and performance. The    |
|                       |      |        |                | company is headquartered in    |
|                       |      |        |                | Vancouver, British Columbia.    |

1The standard abbreviation is NDS, but an additional N can be added to emphasize that this is Nintendo's Nintendo DS, not another company's Nintendo DS.
**Console** | **Code** | **Symbol** | **Exchange** | **Trades as**  
---|---|---|---|---  
PlayStation (original) | PSX | PSX | NYSE | Phillips 66  
PlayStation Portable | PSP | NYSEARCA | Invesco Global Listed Private Equity ETF | Nuverra Environmental Solutions Inc.  
Nintendo Entertainment System | NES | NYSEARCA | Nuverra Environmental Solutions Inc.  

PlayStation began as a CD-ROM expansion for the SNES! But then Nintendo was like j/k we are going to make one with Philips instead! But then Nintendo was like j/k also about that, and made the Nintendo 64. Philips went on to release the abysmally bad console called CD-i, making full spiteful use of their contractual rights to Nintendo characters with abysmally bad titles like *Hotel Mario*. Sony went on to make the PlayStation, mostly for revenge.

<table>
<thead>
<tr>
<th>Console</th>
<th>Code</th>
<th>Symbol</th>
<th>Exchange</th>
<th>Trades as</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sony PlayStation 3</td>
<td>PS3</td>
<td>PS3</td>
<td>FWB (Frankfurt)</td>
<td>Agilysys Inc.</td>
</tr>
<tr>
<td>Sony PlayStation 4</td>
<td>PS4</td>
<td>PS4</td>
<td>FWB (Frankfurt)</td>
<td>Phoenix Solar Aktiengesellschaft</td>
</tr>
<tr>
<td>PlayStation Portable</td>
<td>PSP</td>
<td>PSP</td>
<td>NYSEARCA</td>
<td>Nuverra Environmental Solutions Inc.</td>
</tr>
</tbody>
</table>

Sony went on to make the PlayStation, mostly for revenge.

The X in “PSX” stands for the × symbol in the PlayStation’s official occult incantation: \( \triangle \circ \times \square \). The 3 in “PS3” stands for the other three symbols. This console followed the PlayStation 2 and was released in 2006. It’s sort of like the PlayStation 2 but moreso.

Well, what do you know: They keep making PlayStations. In fact, the PlayStation 5 was announced in 2019, but is still privately traded; it is expected to IPO in late 2020.

Unique for using optical discs for storing its games, this 2005 portable disc-man was fairly successful. It is technically more powerful than the contemporaneous Nintendo DS, but ultimately sold 80 million fewer units than it.

This grey 8-bit family computer from 1985 was the breakthrough console for Nintendo, before we even knew that we would have to keep getting consoles every few years. Several Nintendo franchises were born here: Zelda, Metroid, Kirby, Punch-Out!!, and Wii Fitness.
### Console Code Symbol Exchange Trades as

<table>
<thead>
<tr>
<th>Console</th>
<th>Code</th>
<th>Symbol</th>
<th>Exchange</th>
<th>Trades as</th>
</tr>
</thead>
<tbody>
<tr>
<td>Super Nintendo Entertainment System</td>
<td>SNES</td>
<td>SNES</td>
<td>NASDAQ (New York)</td>
<td>Senestech Inc.</td>
</tr>
<tr>
<td>At the time this console was released in 1990, it was believed that all progress followed a trajectory consisting of ( X ), Super ( X ), Mega ( X ), Hyper ( X ), Giga ( X ), Ultimate ( X ), and then ( X ) Infinite. This was eventually disproved by its successor, the Nintendo 64.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Nintendo GameCube</th>
<th>GCN</th>
<th>GCN</th>
<th>TSXV (Toronto)</th>
<th>Goldcliff Resource Corporation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technically a rectangular prism, the GameCube is a tiny-disc-based system that followed the Nintendo 64. It was released in 2001.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Nintendo Switch</th>
<th>NS</th>
<th>NS</th>
<th>NYSE (New York)</th>
<th>NuStar Energy L.P.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Following the relatively unsuccessful and confusingly-named Wii U, Nintendo somehow made itself quite relevant again with this hybrid home/portable console (this is perhaps to what the “Switch” refers). Released in 2017, its competition includes powerful boxes such as the XBox One and PS4.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>WonderSwan Color</th>
<th>WSC</th>
<th>WSC</th>
<th>NASDAQ (New York)</th>
<th>Willscot Corporation</th>
</tr>
</thead>
<tbody>
<tr>
<td>The WonderSwan Color is a hand-held gaming console released by Bandai in 2000. It followed the WonderSwan (makes sense) and preceded the SwanCrystal (??). The console was modestly successful in Japan with about 100 games, but ultimately lost in popularity to Nintendo’s Game Boy Advance.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>XBox One</th>
<th>XONE</th>
<th>XONE</th>
<th>NASDAQ (New York)</th>
<th>ExOne Co.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft is expert at creative counting. We have Windows: 1, 2, 3, 3.1, 3.11 For Workgroups, 95, CE, 98, 98b, NT, ME, 2000, XP, Server 2003, Vista, 7, 8, 8.1, 10. For XBox: XBox, 360, One, One S, One X, Series X (which will be called simply “XBox”). Only the XBox One is publicly traded.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 1:** Isomorphism between video game consoles (left columns) and stock market (right columns).

---

2There is not consensus on this abbreviation; Wikipedia uses XBO for example.
Figure 2: Market excitement on the day the experiment ends! The contract at the top (Euro–US Dollar exchange rate) is well-behaved, with a bid–ask spread of half a basis point. Ginger Beef Corp GB’s spread is nuts: It is the difference between a total valuation of $1.13M and $2.54M Canadian. Genesis Healthcare has unlocked an achievement: Trading was halted with the last sale at $1.44 (H1.44) due to its price dropping so much.

4 A watched dollar never bills

The next step of the process is to wait it out. If you wish to follow along, place an opaque sheet of paper over the time-series in Figure 4 and move it at the desired pace, revealing information about the performance of each investment. Major video-game events are labeled on this figure, so you can also do a spit-take in real time as decade-tenured Nintendo CEO Reggie Fils-Aimé steps down and is replaced by a man whose name is really... Doug Bowser?

5 Putting my mouth where my money is

Finally, after a year of investing, it is time to divest and reap the monetary harvests. Selling stocks is the simple inverse of buying them.

One surprise is that on March 9th, the first trading day after one year of holding, the stock markets were very active (I think that this was excitement due to an announcement of a new type of beer from the Corona beverage company?). Two issues arose (Figure 2):

- The bid–ask spread became quite wide. There are really two prices for a security: The “bid” (the highest amount that someone is currently willing to pay to buy it) and the “ask” (the lowest amount that someone is willing to sell it for). Gamers can perhaps think of this like the “new” vs. “used” price for a console. When the market is functioning correctly, these prices are typically within a few cents of one another. The issue was even worse for symbols that have low volume (number of trades/day).
For example, GB had a bid of 0.085 CAD and an asking price of 0.190—more than twice the price! Since I tried to get a favorable price when selling the portfolio, this delayed some divestment for several days.

- Due to much market excitement, trading was halted for many securities! This happens automatically on some exchanges when the price changes more than some amount (say, 15%) in one day, in order to prevent “flash crashes” (this is where an SSD drive fails and loses important banking data). Halted trading of course also delays divestment.

The final balance is displayed in Figure 3; these are the prices actually paid or proceeds actually received.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Date Bought</th>
<th>Num @ Price</th>
<th>Cost Basis</th>
<th>Date Sold</th>
<th>Num @ Price</th>
<th>P&amp;L (USD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>XONE</td>
<td>3/8</td>
<td>10 @ 9.302</td>
<td>93.02 USD</td>
<td>3/9</td>
<td>10 @ 5.1927</td>
<td>($41.09)</td>
</tr>
<tr>
<td>WSC</td>
<td>3/8</td>
<td>10 @ 9.97</td>
<td>99.70 USD</td>
<td>3/9</td>
<td>10 @ 13.17</td>
<td>$32.00</td>
</tr>
<tr>
<td>SNES</td>
<td>3/8</td>
<td>100 @ 1.0973</td>
<td>109.73 USD</td>
<td>3/9</td>
<td>5 @ 2.05</td>
<td>($99.48)</td>
</tr>
<tr>
<td>PSX</td>
<td>3/8</td>
<td>1 @ 94.1073</td>
<td>94.10 USD</td>
<td>3/9</td>
<td>1 @ 65.601</td>
<td>($28.50)</td>
</tr>
<tr>
<td>PSP</td>
<td>3/8</td>
<td>9 @ 11.1273</td>
<td>100.15 USD</td>
<td>3/9</td>
<td>9 @ 10.395</td>
<td>($8.60)</td>
</tr>
<tr>
<td>NS</td>
<td>3/8</td>
<td>4 @ 26.88</td>
<td>107.52 USD</td>
<td>3/9</td>
<td>4 @ 16.29</td>
<td>($42.36)</td>
</tr>
<tr>
<td>NDSN</td>
<td>3/8</td>
<td>1 @ 131.2673</td>
<td>131.27 USD</td>
<td>3/9</td>
<td>1 @ 134.64</td>
<td>$3.37</td>
</tr>
<tr>
<td>GG</td>
<td>3/8</td>
<td>10 @ 10.62</td>
<td>106.20 USD</td>
<td>3/9</td>
<td>3 @ 49.16</td>
<td>$41.28</td>
</tr>
<tr>
<td>GEN</td>
<td>3/8</td>
<td>75 @ 1.36</td>
<td>102.00 USD</td>
<td>3/9</td>
<td>75 @ 1.415</td>
<td>$4.13</td>
</tr>
<tr>
<td>NES</td>
<td>3/8</td>
<td>11 @ 9.4973</td>
<td>104.47 USD</td>
<td>3/9</td>
<td>11 @ 2.53</td>
<td>($76.64)</td>
</tr>
<tr>
<td>GCN</td>
<td>3/8</td>
<td>1000 @ 0.08</td>
<td>80.00 CAD</td>
<td>3/9</td>
<td>1000 @ 0.12</td>
<td>$29.29</td>
</tr>
<tr>
<td>GB</td>
<td>3/8</td>
<td>500 @ 0.185</td>
<td>92.50 CAD</td>
<td>3/12</td>
<td>500 @ 0.085</td>
<td>($37.25)</td>
</tr>
<tr>
<td>PS3</td>
<td>3/12</td>
<td>5 @ 18.29</td>
<td>91.45 EUR</td>
<td>3/9</td>
<td>5 @ 22.2</td>
<td>$24.23</td>
</tr>
<tr>
<td>PS4</td>
<td>3/12</td>
<td>3000 @ 0.0267</td>
<td>80.10 EUR</td>
<td>3/10</td>
<td>3000 @ 0.013</td>
<td>($45.46)</td>
</tr>
</tbody>
</table>

**Total:** 5 NaN 1366.34 USD 91/20 NaN ($243.08)

Figure 3: Actual transaction data from the experiment. Each security has a date on which some quantity (around $100 USD) was bought (“Cost Basis”) and a date on which the entirety was sold for some consequent profit or loss (P&L). Note that the number bought is not always equal to the number sold. In the case of SNES, there was a 20:1 reverse split (reverse splits happen to keep the price from getting embarrassingly low: It would be 10.2¢ in this case). GG was acquired, so the sale was actually of a different symbol (NEM). The P&L column and totals have been converted to USD using historic forex rates. Prices omit transaction fees, which are significant especially for non-US exchanges.

### 6 Results

Most of these investments lost money, as did the overall portfolio. Both the SNES and NES performed very badly, which was a disappointment to me, since these are my two favorites. But this is exactly why this approach is needed: It removes the emotional
component. Both of these fell relentlessly throughout the year and lost more than 90% of their value. The best overall were the Game Gear (GG) and WonderSwan Color (WSC) with PlayStation 3 and GameCube also performing well.

It is interesting how little the game consoles react to major video game news (Figure 4). The announcement of the PlayStation 5 in October didn’t seem to cause any movement; this is especially odd for the PlayStation 4 whose value is normally quite volatile. One hypothesis is that this information was already “priced in.” October was when the name “PlayStation 5” was officially unveiled; the successor to the PlayStation 4 had been rumored for some time, but its name was a complete mystery. Arguably, only the cancelation of GDC had any real effect on prices.3

3Also the event marked “Sale”, which is when the experiment ended. It seems like a joke that this would affect the market, but this is literally true for extremely low-volume stocks where my transaction may have been a significant portion of the day’s activity. You can tell a low-volume stock from long flat horizontal lines.

Figure 4: Relative daily closing price for the fourteen publicly-traded video game stocks in the experiment. The closing price is adjusted in arrears for dividend payments and splits. Each price is 1.0 on the date of acquisition (9 March 2019) and other closing prices are given relative to this value. For example, GCN was briefly worth more than twice its starting price at the end of September.
7 Is this the most expensive SIGBOVIK paper?

Probably not.

8 Are video games a good investment?

The efficient market hypothesis tells us that every contract is priced according to the true value of the underlying good. In this sense, every transaction is value-neutral, and money can only be made or lost by predicting the future value of goods. With no information about the thing being bought, as is the case for a basket of random stocks, the investment should technically not gain or lose money (in expectation).

Of course, we do know another theorem about the stock market, from economics: The market always goes up in the long-run! The US Government considers a year to be “long term” for the sake of capital gains tax, so we should expect that randomly selected stocks increase in value over this long term.

So why, in fact, did this investment lose 17.8% of its value? This is because the stocks are not randomly selected; they represent video game consoles. Because video games are high-tech items which rapidly become out of date, they actually tend to depreciate with time. Only a few beloved classic consoles gained in value, for nostalgia reasons.

So this finally allows us to answer the question: What is the best game console? And the answer, established with an elegant isomorphism:

All video game consoles are bad.

References


Reviewer: Count Dracula, Sesame St, Cyberspace
Rating: Great execution of absurd idea.
Confidence: Based on counting irrelevant but objective things like vowel pairs in the paper, absurdly high. But measures like that are not confidence-inspiring, so on balance, low.

Masterful job of optimizing an “objective” measure by gaming the data.

This is a poster-child for the application of Goodhart’s Law and the vacuousness of so-called “objective” rankings that blindly count stuff – like h-index and impact factors.
Funny Business

38 Can a paper be written entirely in the title? 1. Introduction: The title pretty much says it all. 2. Evaluation: It gets the job done. However, the title is a little long. 3. Conclusion: Yes.

Daniel Gaston

Keywords: innovation, strength, progress

39 Erdős-Bacon-Sabbath numbers: *Reductio ad absurdum*

Maria Klawe *et al.* (incl. Harry Q. Bovik!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!)

Keywords: Paul Erdős, Kevin Bacon, Black Sabbath
Can a Paper Be Written Entirely in the Title? 1. Introduction: The Title Pretty Much Says it All. 2. Evaluation: It Gets the Job Done. However, the Title is a Little Long. 3. Conclusion: Yes.

Daniel Gaston
University of Delaware
Paper 9: Can a paper be written entirely in the title? 1. Introduction: The title pretty much says it all. 2. Evaluation: It gets the job done. However, the title is a little long. 3. Conclusion: Yes.

Reviewer: Skirt Steak
Rating: 100%
Confidence: Kim Possibly Confident

While the immediate academic implications of this paper are deep and far-reaching, a more thorough analysis reveals that it also has carefully hidden, previously undiscovered lore on the hit American television series Cory in the House, featuring Kyle Massey and Jason Dolley.

The first tip-off is found even before reaching the paper itself in the author’s pseudonym, which is a pseudonym for ”Cory Baxter”. This is a subtle nod to Cory Baxter, the main character of Cory in the House.

In the second paragraph, we find the real hidden jewel of lore. By taking the first letter of every sentence in this paragraph and placing them in reverse order, a working URL is revealed. Each time the URL is opened, it has a 50% to redirect to either of two pages: Webkinz.com, and the Wikipedia page for Jeff Bezos. It is a little known fact that Cory’s favorite game is Webkinz. It follows that Cory’s favorite multibillionaire American internet and aerospace entrepreneur, media proprietor, and investor is Jeff Bezos.

An extra nugget of lore is planted in the final paragraph, in the last sentence, in the last letter. By examining the lower edge of the bottom right serif with an electron microscope, you will find Cory himself. This has huge implications for the hit American television series Cory in the House, featuring Kyle Massey and Dan Schneider. The only explanation for this observation is that Cory is only slightly taller than a hydrogen atom, and the entire series had to be recorded by shrinking down the rest of the cast and set to an atomic size.
The effort required to make these details subtle yet discernible suggests that the author intended to communicate them secretly to some as-of-yet unidentified group or individual. But the public deserves to hear the truth.
Abstract

A small Erdős number – the “collaborative distance” of authorship between oneself and Paul Erdős – has long been a source of pride for mathematicians, computer scientists and other geeks. Utilizing similar collaborative distance metrics, a small Bacon number (the degree of separation from Kevin Bacon) has been a source of pride for actors, while a small Sabbath number (the degree of separation from Black Sabbath) has been a source of pride for musicians. Previous research in Erdős-Bacon number minimization has reduced the Erdős number of a number of computer scientists to two, which is believed optimal, although the reduction of the Bacon number to four was clearly suboptimal. We extend and improve on this previous work to provide a Erdős-Bacon-Sabbath number minimization that is believed to be close to optimal in all axes.

Derivative Introduction [2]

Paul Erdős co-authored nearly 1500 papers (until his death in 1996), working with nearly 500 collaborators achieving the status of the most prolific mathematician in modern times [4]. Mathematicians thus humorously defined Erdős numbers. A person’s Erdős number is the distance between that person and Paul Erdős in the academic paper collaboration graph [3]. Succinctly, Paul Erdős is the unique person with Erdős number zero; all of Erdős’ immediate co-authors have Erdős number one; in general, if you publish an academic paper with a collaborator who has Erdős number $N$ and none of your other co-authors has Erdős number less than $N$, your Erdős number is $N + 1$.

A similar Bacon number [16] has been proposed for actor Kevin Bacon, except using collaborations in film instead of collaborations in academic papers. Likewise, a similar Sabbath number [12] has been proposed to connect to the members of
the musical group Black Sabbath, using collaborations in musical performances.

Erdős-Bacon-Sabbath numbers were subsequently defined [5] to be the sum of each person’s Erdős, Bacon, and Sabbath numbers.

There is a long tradition of posthumous publication [7], and authors claiming to have collaborated with Erdős have brought his total number of known publications to 1525, his collaborator count to 511, and the Erdős number of the chutzpah-bearing mathematician to one. The latest publications co-written with Paul Erdős appeared more than ten years after his death. With additional rumored works in progress, Erdős’s publication list is expected to grow. In fact, Paul Erdős himself has published a solo work 15 years after his death [8].

In this paper, we describe and demonstrate a technique called Erdős-Bacon-Sabbath Number Minimization.

Rules of the Games

The rules of Erdős number calculation are clear: author a paper in a peer-reviewed publication, either with someone connected by co-authorship with Paul Erdős, or with Paul Erdős himself (the latter being unlikely unless pre-demise work is used in a posthumous publication, or if you are better a communicating with the dead than Edgar Cayce). Simply putting Paul Erdős’ name on your paper does not count (and changing your name to Paul Erdős for purposes of publication is definitely cheating).

The rules of Bacon number calculation [16] are also clear: act in or be otherwise credited in a film with someone connected by film credit with Kevin Bacon¹, or with Kevin Bacon himself. We tried contacting Kevin Bacon, but his agent refused to put us in touch. We contemplated inserting a fair-use clip of Kevin Bacon from an unrelated movie, but knew that would be cheating (but we did think of it).

Finally, the rules of Sabbath number calculation [12] are also clear: connections between a musician and a band or solo artist can only be made if they actually performed or recorded together. However, “session musicians” are valid connections, so musicians who perform live or record with an artist, but are not strictly committed to that band are valid. A new recording [11] can serve as adequate proof, but singing along to a Black Sabbath record is cheating. We also contemplated contacting Ozzy Osbourne, but he postponed the 2019 tour that would have taken him through the Pittsburgh area, and then cancelled the tour altogether.

The theoretically achievable absolute minimum Erdős-Bacon-Sabbath number is two: if Kevin Bacon (who has a Bacon Number of zero) were to become a member of Black Sabbath (thus receiving a Sabbath number of zero), and was to publish a paper with a person with an Erdős number of one (since Paul Erdős is dead, and ineligible as a co-author). The practically achievable minimum Erdős-Bacon-Sabbath number is four (by authoring a paper with a someone with an Erdős number of one, appear in a movie with Kevin Bacon, and perform with a member of Black Sabbath), but the realistically achievable minimum is somewhat higher than that.

¹ The Internet Movie Database http://www.imdb.com is typically used for verification of film/video/YouTube credits.
Computation of Erdős, Bacon, and Sabbath Numbers

In [1], Maria Klawe co-authored with Paul Erdős and has an Erdős number of one, thus guaranteeing that all authors on this paper have an Erdős number no greater than two.

Daniel V. Klein has a Bacon number of 2, having appeared in [9] with Steve Guttenberg, who appeared in [10] with Kevin Bacon. Mike Ancas also has a Bacon number of 2, having appeared as an extra in [17] with Tom Hanks, who appeared in [18] with Kevin Bacon. All authors on this paper also appear in the documentary about the writing of this paper [11], and thus have a Bacon number no greater than three.

Additionally, Mike Ancas also has a Sabbath number of 1, having been a member of the Bloomsburg PA High School rock band The Rubber Band. In 1971 The Rubber Band performed as one of several warm-up acts for Ronnie James Dio (who in 1979 became the lead singer for Black Sabbath). The authors of this paper have recorded a special musical piece, composed specifically for this paper, and captured in [11]. Their Sabbath number is therefore no greater than two.²

The combined Erdős-Bacon-Sabbath number for all the authors of this paper is therefore no greater than seven, surpassing the rarified company of the only three other people hitherto known to have an Erdős-Bacon-Sabbath number of eight: Stephen Hawking, Ray Kurzweil, and Daniel Levitin, a professor of psychology and behavioral neuroscience at McGill University [6]. They now are tied with the previous record-low Erdős-Bacon-Sabbath number (held by Lawrence Krauss [13]) with an Erdős-Bacon-Sabbath number of seven!

Not to brag or nothin’, but Maria Klawe (E=1, B=3, S=2) and Daniel V. Klein (E=2, B=2, S=2) have now beaten that record with an Erdős-Bacon-Sabbath number of six. Finally, with a near-optimal EBS number minimization, Mike Ancas (E=2, B=2, S=1) has algorithmically achieved an incredible Erdős-Bacon-Sabbath number of five!

Sheet Music & Documentary

You’ve read the paper, now read the music (in the appendix)! Lastly, you get to watch the documentary at [11].

Acknowledgements

Thanks, Mom! Thanks, Dad! Thanks to spouse(s) and/or partner(s), significant other(s), groupies, and random tinder dates!

References


² Early research on low Sabbath Numbers included DK Fackler, who has a Sabbath number of 2. DK performed with Roger Daltrey [14] in 1994, who in turn performed in Wembley Stadium with “The Who” at Live Aid (1985), where Black Sabbath also performed [15].


Appendix: “Erdös Bacon Sabbath Number Reduction” in A Maj
Daniel V. Klein, op 2+2+3