The Association for Computational Heresy

presents

A Record of the Proceedings of

SIGBOVIK 2014

The eighth annual intercalary robot dance party in celebration of workshop on symposium about Harry Q. Bovik's 2^6th birthday

Carnegie Mellon University

Pittsburgh, PA

April 1, 2014

Association for Computational Heresy

Advancing computing as Tomfoolery & Distraction

SIGBOVIK

A Record of the Proceedings of SIGBOVIK 2014

ISSN 2155-0166

April 1, 2014

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

SIGBOVIK 2014

Message from the Organizing Committee

Multiple-Choice Section

1. *5 points.* Please indicate below the emotions with which the SIGBOVIK Organizing Committee presents this document. Circle up to three, but no fewer than four, choices.

   (a) Joy

   (b) Embarrassment

   (c) Surprise

   (d) Admiration

   (e) Arrogance

   (f) Pride

   (g) Terror

2. *5 points.* Which is the correct full name of this conference?

   (a) The Eighth Annual Intercalary Conference about Symposium on Human Dance Party of Workshop in Celebration of Harry W. Bovik's 2^6th Birthday.

   (b) The Eightieth Annual Intracalary Workshop about Workshop on Robot Dance Party of Conference in Mourning of Harry Q. Bovik's 2^6th Birthday.

   (c) The Eighth Centennial Intercalary Workshop about Symposium on Robot Karaoke Party of Conference in Celebration of Harry Q. Bovik's 3^6th Birthday.

   (d) The Eighth Annual Intercalary Workshop about Symposium on Robot Dance Party of Conference in Celebration of Harry Q. Bovik's 2^6th Birthday.

Reading Comprehension Section

25 points.

This year's SIGBOVIK marks our eighth continuous year of high-quality research in a rich variety of topics, including but not limited to applied phlogistonics, synergistic hyperparadigmatism, and elbow macaroni. Eight is a very special annuality of this conference, as it not only breaks the

"highest annuality" record set last year, but also is the fourth power of two number of years for which this conference has been in existence. If you're not impressed yet, note that the fourth power of two is a very special power of two, as four is the second power of two, and you know what they say about the second power of two.

As it is a very special year, we have decided to conduct a survey of past SIGBOVIKs, or SIGBOVIX for short, classifying the various papers to get a feel for where future SIGBOVIKs might be headed. This was driven by the realization depicted in Figure 1.

	Serious treatment	Humorous treatment
Serious idea	**Mainstream conferences**	**SIGBOVIK**
Humorous idea	**SIGBOVIK**	**SIGBOVIK**

Figure 1: A research paper is comprised of an *idea* and a *treatment* of that idea. Each of the idea and the treatment may be either serious or humorous.

Observe that SIGBOVIK offers a venue for three times as many different types of research as "mainstream" conferences. We won't belabour the obvious conclusion that SIGBOVIK is superior, although, you know, just getting that out there, we were all thinking it; rather, we are interested in distinguishing among SIGBOVIK's three major categories of research.

1. **Humorous idea, humorous treatment.** The most common, but by no means the most lowly, of SIGBOVIK publications. The research contribution is typically contained entirely within the paper itself; no separate artefact is constructed.

2. **Serious idea, humorous treatment.** The rarest specimen of SIGBOVIK research. Often a retelling of famous events, people, or theoretical results from mainstream computer science.

3. **Humorous idea; serious treatment.** Frequently denoted by independent artefacts accompanying the paper submission, such as a website, compiler implementation, hardware, or proof. Such publications would often be suitable for acceptance at "mainstream" conferences, were the core idea not out of scope.

~~I~~ We surveyed the proceedings of past SIGs BOVIK ~~the night before finalizing the proceedings and sending them off to Lulu~~ like responsible researchers. Our methodology is definitely completely immune to any biases that might be caused by the survey not being blinded in any way or by it being conducted by only one person, and *definitely* free from any bias related to the study subject knowing in advance which conclusions would be drawn. The results of our survey are shown in Figures 2 and 3 ~~below~~ on the next page.

	SIGBOVIK year of incidence							
	2007	2008	2009	2010	2011	2012	2013	2014
Humorous treatment of humorous idea	32	20	30	33	16	17	15	16
Humorous treatment of serious idea	2	0	1	0	0	1	2	0
Serious treatment of humorous idea	2	4	3	0	4	6	6	8
Other	1	1	5	3	0	2	2	0
Total publications	37	25	39	36	20	26	25	24

Figure 2: Results. The 'other' category comprises submissions that defy my perfect classification scheme, including cryptic iconography, video games, comics, choose your own adventures, and take-home midterm examinations.

	SIGBOVIK year of incidence							
	2007	2008	2009	2010	2011	2012	2013	2014
Meta-humor about research	7	2	8	8	8	1	2	2
Type theory or programming jokes	8	3	1	7	2	1	2	4
Puns	4	4	8	1	0	0	2	3
Poop or dick jokes	1	0	1	1	0	1	1	2
Politics, economics, or patents jokes	0	0	0	0	1	2	2	1
Social media jokes	0	0	0	2	2	1	0	0
Other	12	11	11	12	3	11	4	4

Figure 3: Breakdown among category-1 publications of subject of humor. Among this new 'other' category dwell comestibility theory, NP-completeness, cat pictures, computational impossibilities, the supernatural, and robot uprisings and other apocalypse situations.

A number of trends are evident. First and foremost is the decline over time of the proportion of category-1 submissions and a corresponding increase in submissions accompanied by an actual artefact that must have been put together with blood, sweat, and so forth, rather than just the cushy endeavour of writing down shallow drivel in LaTeX and calling that a "research paper". Go us! Just kidding. We value all SIGBOVIK submissions highly, ~~except for~~ especially yours. But really.

Second, we note the prominence among category-1 submissions of meta-research papers; that is, papers about writing papers, giving talks, or improving productivity. It is unsurprising that this is a popular subject for obvious reasons. Finally, we note a modest decline in the proportion of programming language papers, which we attribute to SIGBOVIK's audience's interests expanding beyond the core of its type-theoretician founding mothers and fathers, to include such newfangled topics as Twitterers and Bitdollars.

Without further ado, adon't, or adonuts, the proceedings of your SIGBOVIK #8.

You Won't Believe This Table of Contents

Track 88,888: Improve Your Own Research With These Five Weird Tricks

Track 888,888: These New Ways to Program Will Blow Your Mind

Track 8

How to Keep Adults Busy and Other Fun Games for Children

1. **How to Keep a Graduate Student Busy**
   Paul Stansifer
   Keywords: all, work, and, no, play, makes, paul, a, dull, boy

2. **New Results in k/n Power-Hours**
   Dr. Tom Murphy VII Ph.D.
   Keywords: generalized binge drinking, maths, finite-state automata, abstract interpretation

3. **Linear Logic Free**
   Carlo Angiuli
   Keywords: linear logic, free to play, app store

4. **It Still Seems That Black Has Hope in These Extremely Unfair Variants of Chess**
   Dr. Tom Murphy VII Ph.D., Ben Blum, and Jim McCann, in italics ..... oh! .... nervous laughter .... no, that's not part of the name still. I stopped saying the name. Why are you still typing? This isn't the name any more. End quote.
   Keywords: chess, stratego, guess who, clue, candy crush saga, game-tree search, combinations

How to keep a graduate student busy

Paul Stansifer

1 April, 2014

Abstract

We propose a method for indefinitely occupying certain graduate students that requires only a finite amount of input information.

1 Motivation

Some graduate students are incapable of understanding recursion. Such students need to be kept busy, lest they interfere with all of the actual real work that is, in fact, really happening.

2 Method

To achieve this, simply apply the method described by Stansifer [1], and iterate by one step more.

3 Mandible

You're not done with the previous section yet. Get out of here!

References

[1] Paul Stansifer. How to keep a graduate student busy. *SIGBOVIK*, page [can I get back to you on that?], 2014.

SIGBOVIK 2014 Paper Review

Paper 18: How to keep a graduate student busy

Norman I. Strong, RSU State University
Rating: ?? (??)
Confidence: ??/4

Program Chair's note: unfortunately, we are unable to print a review for this paper as the reviewer assigned to the submission did not finish reviewing the paper by the deadline. The last comment received from the reviewer was "Since the 2014 work of Stansifer is so vital to your method, please summarize the paper inline rather than assuming the reader is already familiar it, as it is time-consuming and tiresome for the reviewers to locate and review the prior work." The PC apologizes for the delay and will pass on the rest of the review when it is submitted.

New results in $^k/_n$ Power-Hours

Dr. Tom Murphy VII Ph.D.*

1 April 2014

Abstract

We correct for inebriated missteps, using computational methods to establish new bounds in generalized $^k/_n$ Power-Hour theory.

Keywords: generalized binge drinking, maths, finite-state automata, abstract interpretation

Introduction

A 2012 paper by Blum, Martens, Murphy, and Lovas[1] introduced the $^k/_n$ Power-Hour, a fractional variant on the well-known drinking game. In a traditional Power-Hour, participants drink one shot of beer per minute for 60 minutes. Since 5–6 beers in an hour sometimes have adverse effects, some players opt for an attenuated version of the game wherein fewer than 60 shots are consumed. However, since the game is frantic and played simultaneous with others, it is critical to have a mechanical procedure for performing the attenuated Hour. The framework by Blum *et al.*, hereafter BMML, gives a handful of simple operations that can be used to define a state machine among p players:

- At the beginning of each minute, each player has at most one shot glass in front of him or her
- The shot glass must be in one of three states: Filled $\uplus$, empty $\cup$, or overturned $\cap$
- Atomically, each player performs an action based only on the state of his or her cup. If not in possession of a cup (written $\uplus$), the only action is to do nothing. With a cup:
  - The player may drink $\overset{+}{\Rightarrow}$, or not drink $\Rightarrow$
  - The player may pass the cup in any state to any player (a fixed player per action)
  - However: If the cup is filled and the player did not drink, it must be passed in the filled state
  - A player may not receive more than one cup in the same round

Every assignment of rules and starting condition to p players yields a deterministic outcome, though some of these are illegal (because they result in two or more cups being passed to the same player in some round). For legal games, the outcome is that the p players have consumed k_i shots of beer where $1 \leq i \leq p$ and $0 \leq k_i \leq 60$. For the traditional power hour, the player starts with an empty cup, at each step drinks,[1] leaves the cup empty, and passes to herself.

While the authors made a mostly clear definition of BMML and presented some initial results, these results contained multiple serious errors and the paper abruptly switches notation and assumptions several times, and rambles incoherently. By their own admission, the authors were drinking while they wrote it, taking only one hour to do so. Don't drink and derive, kids!

This paper revisits the problem of BMML from a modern, sober perspective, clarifies some of the original results, and presents several new ones and a few conjectures. It is based on several pieces of software, whose source is available online.[2]

1 One-player $^k/_n$ Power-Hours

The goal of the $^k/_n$ Power-Hour is to attenuate the number of drinks consumed by the p players, and its expres-

[1] In practice, this is done by filling the cup and then drinking it.

[2] `http://sourceforge.net/p/tom7misc/svn/HEAD/tree/trunk/powerhour/`

sive power comes from the ability to encode some state in the orientation of the cups, and propagate that state via passing them from player to player. Even without passing cups, the ability for a single player to attenuate his drinking is nontrivial. Playing drinking games alone is sad indeed, but the solo $^k/_n$ Power-Hour still has practical applications. When playing a Power Hour with others, if each player's desired k is attainable through solo methods then there is no need for passing cups, which simplifies the ergonomics considerably. A common case is where some of the players would like to do half–Power-Hours, which is easily achieved in BMML by transitioning $\cup$ to $\uplus$ without drinking and $\uplus$ to $\cup$ by drinking, and passing to oneself.[3]

A full list of attainable $^k/_n$ Power-Hours where $p = 1$ appears in Figure 1. Possible values of k are $\{0, 1, 2, 20, 29, 30, 31, 40, 58, 59, 60\}$. The BMML paper claimed that the possible values were $\{0, 1, 2, 20, 30, 40, 58, 59, 60\}$, describing 31 for example as "super impossible." Achieving 31 is somewhat interesting. One way to do it is to start with $\cup$, and use the rule that $\cup$ means drink and then fill the cup. We then use the rule that $\uplus$ means drink and flip the cup, and $\cap$ means don't drink and fill the cup. Essentially we use $\cup$ to mean "this is the very first state" and then take shots on alternating minutes by using $\cap$ and $\uplus$ to encode the parity. Exploiting non-steady-states like this (Figure 1) is how we achieve k that does not share many factors with n.

It is tractable to work out the possibilities for the solo case by hand, though apparently not while drinking [1]. These results were generated by a computer program, which is probably necessary for $p > 1$. In the remainder of the paper, I'll describe several different approaches for exploring this space, and generalizations of it, computationally.

2 Two-player $^k/_n$ Power-Hours

For more players, the number of possible configurations explodes. Let's make the following definitions to bound the size:

- $t = 4$, the number of starting states ($\uplus$, $\cup$, $\cap$, $\uplus$)
- $a = 2 \times p \times 3$, the number of actions given a cup. The player can drink or not drink, pass to any player, and in 3 configurations ($\uplus$, $\cup$, $\cap$)

[3]There are many variations, but this was the strategy used many times in practice before being generalized to BMML.

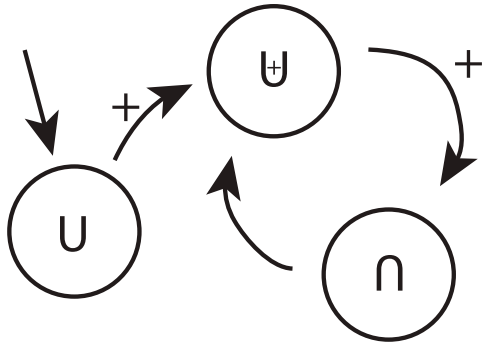


Figure 1: State machine that achieves $k = 31$ in a solo BMML Power Hour. + on an edge means the player drinks. The disembodied incoming edge is the start state. The player always passes to herself.

Then the number of configurations is bounded by $(t \times a^3)^p$. For $p = 1$ this was just 864. For $p = 2$ it is 47,775,744; for $p = 3$ it's 12,694,994,583,552, already beyond the limits of straight enumeration.

However, this is just an upper bound. For one thing, the base of the exponent is actually bounded by

$$t \times a^2 \times a_{\text{filled}}$$

where $a_{\text{filled}} = (p \times 3) + p$ (the actions that can be taken on $\uplus$, where if the player does not drink, then he must pass the cup $\uplus$).

The values for $p \in \{1, 2, 3\}$ are still 576; 21,233,664; 3,761,479,876,608. There are a few other simplifications possible. Many of these games are illegal because they result in multiple cups being passed to the same player in some turn. These are difficult to exclude analytically, but there are some sufficient conditions; for example, if two players pass to the same player no matter their input state, and every player starts with a cup, then their cups always collide. There are also many games that are isomorphic. For one thing, $\cup$ and $\cap$ are not distinguished in the rules at all, so any two configurations where these are simply swapped has the exact same outcome. Likewise for permuting the players.

21 million configurations is no big deal for a modern computer. A simple SML program computes all of the configurations and runs them; ones that are found to be illegal are rejected. (It implements the first simplification having to do with $\uplus$ when generating the configura-

k	start	rules
0	$\uplus$	$\cup\Rightarrow ?$, $\cap\Rightarrow ?$, $\uplus\Rightarrow ?$
1	$\cup$	$\cup\overset{+}{\Rightarrow}\cap$, $\cap\Rightarrow\cap$, $\uplus\Rightarrow ?$
2	$\cup$	$\cup\overset{+}{\Rightarrow}\uplus$, $\cap\Rightarrow\cap$, $\uplus\overset{+}{\Rightarrow}\cap$
20	$\cup$	$\cup\Rightarrow\cap$, $\cap\Rightarrow\uplus$, $\uplus\overset{+}{\Rightarrow}\cup$
29	$\cup$	$\cup\Rightarrow\cap$, $\cap\Rightarrow\uplus$, $\uplus\overset{+}{\Rightarrow}\cap$
30	$\cup$	$\cup\overset{+}{\Rightarrow}\cap$, $\cap\Rightarrow\cup$, $\uplus\Rightarrow ?$
31	$\cup$	$\cup\overset{+}{\Rightarrow}\uplus$, $\cap\Rightarrow\uplus$, $\uplus\overset{+}{\Rightarrow}\cap$
40	$\cup$	$\cup\overset{+}{\Rightarrow}\cap$, $\cap\Rightarrow\uplus$, $\uplus\overset{+}{\Rightarrow}\cup$
58	$\cup$	$\cup\Rightarrow\cap$, $\cap\Rightarrow\uplus$, $\uplus\overset{+}{\Rightarrow}\uplus$
59	$\cup$	$\cup\Rightarrow\cap$, $\cap\overset{+}{\Rightarrow}\cap$, $\uplus\Rightarrow ?$
60	$\cup$	$\cup\overset{+}{\Rightarrow}\cup$, $\cap\Rightarrow ?$, $\uplus\Rightarrow ?$

Figure 2: All the possible k for a solo Power-Hour in BMML. A superscript + means that the player drinks. The symbol ? means that any cup state can be used in that position. Note that 29 and 58 require wasting a shot of beer (the game ends with the shotglass full); all the others but 31 permit a variant where a shot is wasted as well. We do not concern ourselves much in this report with these leftover shots.

tions, since it can be done statically.) All of the possible outcomes are shown as black squares in Figure 3.

Each cell represents a pair of $\langle k_1, k_2\rangle$ for the number of shots imbibed by players 1 and 2. 454 of the $61^2 = 3721$ combinations are achievable. Note that column 0 represents the case where player 1 drinks nothing. It dominates the matrix in the sense that if $\langle k_1, k_2\rangle$ is achievable, then $\langle 0, k_2\rangle$ is as well. Most of the time it is easy to see how this is done: Take the configuration that produces $\langle k_1, k_2\rangle$ and do the same, but player 1 simply performs her actions without drinking. This works except for the case where player 1 receives a $\uplus$ and passes it in a state other than $\uplus$. The player can't simply not drink, as this is illegal (the beer must be emptied, and BMML does not permit such messy reductions). It is curious that this does not affect the result; I discuss this further in Section 7.2. Another interesting column is the last one, which represents outcomes of the form $\langle 60, k_2\rangle$, where the first player achieves a full Power-Hour. This of course includes all of the k_2 achievable solo (the players can just do their thing without interacting). But some new k are now achievable:

Figure 3: All of the possible outcomes (k) for the two players in a BMML Power-Hour. The matrix is symmetric, of course, since the players are interchangeable.

$\{3, 4, 15, 28, 32, 45, 56, 57\}$. Interacting with a player doing a full Power-Hour still affords us a few additional bits of information that can be used to attenuate the other player's consumption. The solution for 45 is instructive, and appears in Figure 4.

This is a useful result, but it may be the case that someone wants to drink exactly 27 shots of beer, which is not possible with just two players in BMML. There are two avenues to explore: Adding more players, and generalizing BMML. We begin with the three-player case.

3 Three-player $^{k}/_{n}$ Power-Hours

With 3.7 trillion possible configurations, enumeration is not feasible. But as we observed before, many of these combinations are illegal (they result in a player recieving two cups), and many are isomorphic to one another. By being clever about how we explore the configurations, testing "all" the three-player configurations becomes feasible.

Here is a one-player BMML configuration that illustrates a particular kind of redundancy:

start $\cup$ $\cup\overset{+}{\Rightarrow}\cup$ $\uplus\overset{+}{\Rightarrow}\cap$ $\cap\Rightarrow\uplus$

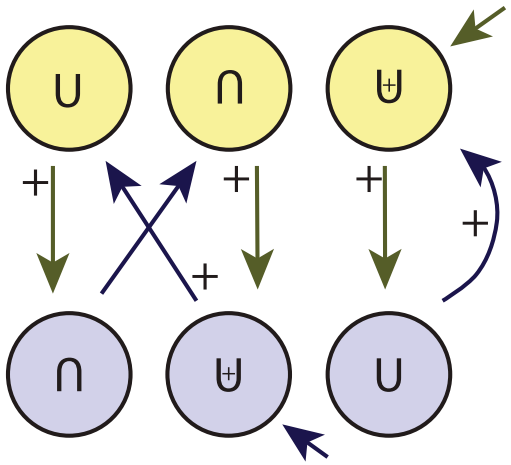


Figure 4: State machine that achieves $\langle k_1 = 45, k_2 = 60 \rangle$ in a two-player BMML Power Hour. The bottom row of states are for player 1, who drinks 45, and the top for player 2, who drinks 60. Clearly, player 2 must drink at every step. The players always pass to each other, with the two cups exchanging hands each turn. The cycles for the two cups are disconnected; one alternates between $\uplus$ in player 2's hand and $\cup$ in player 1's (cycle of length 2), drinking on each turn. This cycle yields 30 drinks for each player. The other cycle is of length 4; player 2 drinks on every step (as we know), and player 1 every 4^{th} step, yielding 15 more drinks for a total of 45.

The cup starts empty, and at each step the player fills it and drinks (traditional Power-Hour). The player also has rules for the case that she observes a full or overturned cup. *It does not matter what these are* because they can never be used. This example is trivial, but there are many ways that the execution of a configuration can be indifferent to some of its content. Another is a two player configuration like

player 1	start $\cup$	$\cup \overset{+}{\Rightarrow} \cap @1$	$\uplus \overset{+}{\Rightarrow} \cap @2$	$\cap \Rightarrow \uplus @1$
player 2	start $\cup$	$\cup \Rightarrow \cup @1$	$\uplus \overset{+}{\Rightarrow} \cap @1$	$\cap \overset{+}{\Rightarrow} \uplus @2$

where the $@n$ notation means to pass the cup in that state to player n. In this case, the first thing the players do is to pass both of their cups to player 1, which is illegal and ends the game. Again, none of the other rules are ever used.

In order to explore what is possible in three-player games, we exploit this redundancy with a technique like abstract interpretation [2]. The start state is always used, so we begin by enumerating all assignments of start states to players. There are only 4^p. Every other rule starts out undetermined, maybe written like this:

start $\cup$ $\cup \overset{?}{\Rightarrow} ?$ $\uplus \overset{?}{\Rightarrow} ?$ $\cap \overset{?}{\Rightarrow} ?$

Now we execute programs as before, and hope that we never encounter a situation where we depend on a rule. If we finish without ever using one of the ? rules, we evaluated a potentially large group of configurations all at once. During the execution of a configuration, if we need to use a rule that is currently marked ?, we explore all of the possibilities for that spot. This is accomplished by a loop that looks like the following (in Pseudo SML):

```
val queue = (* all abstract configurations *)
val results = (* map from (k_1, k_2)
                 to example *)

fun loop nil = (* done *)
  | loop (h :: t) =
      let
        val res = evaluate h
      in
        insert (results, res);
        loop t
      end handle Expand l => loop (l @ t)

fun evaluate config =
   (* ... *)
   case rulefor cup of
      QuestionMark =>
         raise Expand expandedconfigs
    | (* ... *)

val () = loop queue
```

The key trick here for keeping the code under control is to iteratively evaluate the configurations as usual, but if we find a ?, then we abort the current simulation with an exception that carries along the set of configurations that expand the current one in just that position. This wastes some work (and we often need to restart multiple times per abstract configuration), but not much: If a rule is used at all, it is usually used in one of the first few rounds.

With this technique, we can simulate all possible two-player games with just 15,744,259 game-minutes simulated (with naive enumeration it would be 1.2 billion) in less than 2 seconds on a crappy old computer.

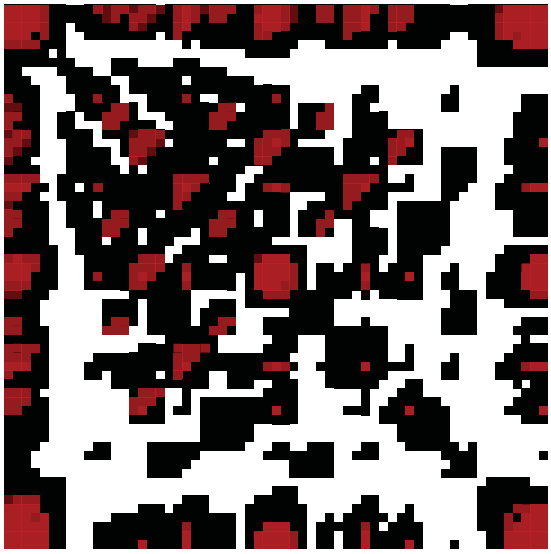


Figure 5: Outcomes possible for the first two players in all different 3-player power hours (black), overlayed by all possible outcomes outcomes for 2-player power hours (red). Mainly included because it looks pretty sweet. The outcomes that are possible with three players are a superset of those with two, which is intuitive: We can add a third player to any game who just does nothing.

It is also feasible overnight to enumerate all three-player games. The results are three-dimensional, of course, but we can display the outcomes for two of the players in the familiar presentation (Figure 5). Note that $\langle k_1, k_2, k_3 \rangle$ is achievable for any $k_1 \in \{0 \ldots 60\}$, $k_2 \in \{0, 1, 2\}$, and some unknown k_3 (projected out of this display). This is a significant improvement over what was achievable in BMML with two players. It suggests that with enough friends willing to follow a program, some set of people are likely to be able to achieve any amount of drinks between them (if they have the ability to construct the right rule set!); see Section 7.2.

The sheer number of configurations for 4 or more players makes these exact enumeration techniques infeasible. However, we have other avenues for generalization (and exploration), which are investigated in the next chapter.

4 Generalized BMML

Like any drinking game, there are several arbitrary things about BMML. While we will not tamper with its essence (for example, allowing beer to be spilled from a cup without drinking it), there are some other variables to adjust. The most naturally flexible is the number of cup states. We will always have ⊎, a cup with beer in it. In BMML we also have ∪ and ∩. But why not ⊃ (cup turned on its side, facing west) or $\overset{\wedge}{\cup}$ (an upright cup with a cocktail umbrella on it)?

We define BMsL, where s is the number of distinct cup states. By convention, the 0^{th} cup state will be the filled cup ⊎ since it has special rules. The remainder will be ⊍ for $i \in \{1, \ldots, s-1\}$. BMML is BM3L where we've just renamed ∪ to ⊎, and ∩ to ⊌.

Clearly, more cups give us more expressive power, and should allow us to reach more outcomes. To illustrate, recall the construction of $k = 31$ in the solo BM3L game (Figure 1). It has a length-2 cycle alternating between two cup states, where the player drinks on every other turn. The third state is just used once as a drinking lead-in to make the total 31 rather than 30. With an additional cup state, we can straightforwardly transform this into a game with an outcome of $k = 32$ by extending the prelude with another state where the player drinks. Of course, the expressive power is not just limited to such extensions; we now can create cycles of new lengths, admit the possibility of more disconnected cycles, and so on.

It is easy to enumerate the 1-player BMsL games. These appear in Figure 6. The interesting range for s is $\{1 \ldots 8\}$.[4] At 8 cup states, the player can achieve any amount in a solo game. Importantly, this extends to any number of players in BM8L, because the players can pass to themselves and not even interact.

The two-player case is much more interesting. We've already enumerated all the possible outcomes for BM3L (Figure 3). It is computationally tractable to enumerate them for $s < 6$. The set of achievable outcomes in BMsL is always contained within BM$(s+1)$L, because we can embed a game from the former into the latter by just never producing the additional cup state and having any arbitrary rule for it. Therefore, we show these results in a composite grid where more and more states are reachable as we increase s (Figure 7).

In order to efficiently enumerate BM5L, I improved

[4] It's not clear that BM0L should be considered legal as the rules speak of a 0^{th} cup, but it is degenerate anyway.

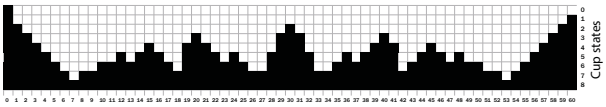


Figure 6: Possible outcomes for BMsL with a single player. The vertical axis shows an increasing number of cup states, and the horizontal axis shows the achievable values of k drinks. With no cup states, it is only possible to drink nothing. By convention, the 0^{th} cup is the special "filled" state, so it is only possible to drink on every turn (60) or never (0). As we add more cup states, the number of achievable states strictly increases; with 8 cup states we can drink any amount. 7 and 53 drinks are the most elusive, and can only be done with 8 cup states.

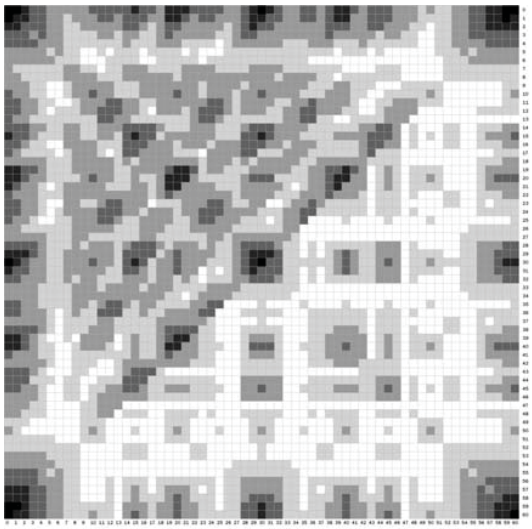


Figure 7: What's achievable for two players in a generalized BMsL game, with s ranging from 1 cup state (darkest) to 5 (lightest).

the algorithm again. Observe that the "drink" action associated with a rule usually does not affect anything but the final outcome. The only exception is that in the rule for $\uplus$, the player must drink if passing in a state other than $\uplus$. Putting this aside for a moment, note that we can just count the number of times each rule was executed for each player, producing an s-dimensional vector $[d_1, \ldots, d_s]$ for each player. That player is able to achieve many different drink totals, specifically, $d_1 \times r_1 + \ldots + d_s \times r_s$ where r_i is 1 if the player should drink on that rule and 0 if not. Simulating a game this way is even more like abstract interpretation (we leave a concrete value free and compute a formula rather than an integer), and allows us to evaluate many concrete games at once. At the end, we simply plug in every legal value for r_i for each player and insert those games into the database. This last step is where we must tend to the exception around $\uplus$. We may not set r_0 to 0 if the player ever passes $\uplus$ in a non-full state. A very close approximation would be to insist that $r_0 = 1$ if the rule in the $\uplus$ position does not output as $\uplus$, but this is inexact, as that rule may never be executed.[5] Instead, during simulation we keep track of whether each player ever actually passed a non-full cup from the $\uplus$ state. If so, then we force $r_0 = 1$, which attends to this special case.

Although this makes earlier enumerations extremely fast and BM5L quite quick, 2-player BM6L ran 26 billion concrete states overnight and made only modest progress. In the absence of fancier techniques for reducing the state space, we must resort to different, inexact approaches.

5 Sampling games

To establish a result like "BM5L cannot achieve $\langle 47, 27\rangle$" we really need to enumerate all the BM5L games. (Or make some ad hoc proof of the fact, which seems quite difficult.) However, to prove an existence result like "BM7L can achieve $\langle 33, 49\rangle$" we only need to have a single example configuration that produces that result. Therefore, we may be able to improve our bounds on what is possible (or generate conjectures) by sampling random configurations.

Sampling is actually much easier than enumeration. There is no need to leave rules abstract. It is also easy to

[5] We may be able to argue that in that case, there always exists another game that does not violate this condition. But I think it is simpler to just implement the rules.

stop and restart because there is no state other than the matrix of what we've found. I use the SML `textformat` library [3] to serialize and deserialize the matrix (which then makes it easy to generate these graphics in a separate program). There are a handful of interesting aspects:

Generating a random configuration. To generate a random game, we can just fill in all of the slots (destination and cup state for each rule, starting cup state) uniformly at random. Many of these will be illegal, but they fail very quickly at runtime; a lazy and pragmatic way to "filter" to legal game. It is not simply a matter of generating all the permutations on $p \times s$ nodes, by the way. Multiple cups can pass through the same player on cycles of different periods, as long as they do not collide within the 60 steps, and acyclic preludes (Figure 1) are important and useful. For a uniformly random 2-player BM7L configuration, 29.23% (measured empirically) are legal. However, we will see later that we do not want to spend so much time exploring configurations where one or both players start without a cup; these are very limited. Therefore, the configuration generator is biased towards producing a cup in the starting states most of the time.

Symmetry. We can get more bang for the buck by considering some obvious symmetries. When a simulation finishes and we have an outcome $\langle k_1, \ldots, k_p \rangle$, it is clear that any permutation of $k_1 \ldots k_p$ is also achievable. We insert every permutation of the drink counts into the database, along with the permuted example configuration. Better still would be to only store the outcomes in some normalized form (e.g. require that $k_1 \leq \ldots \leq k_p$).

We already have exact results for two players in BM5L, so the next uncharted territory is BM6L. The result after apparent convergence appears in Figure 8. The sampling procedure runs for many hours before plateauing overnight with 95.65% of the matrix filled. This suggests that BM6L is not universal for two players, or else the configurations for the missing cells are extremely rare.[6]

This approach scales much better than enumeration and is efficient for all sorts of generalizations (it works best when the dimensionality is low—i.e., two players—and the expressiveness is high—i.e., many cup states).

[6] This is definitely a possibility, as new cells were still appearing after exploring tens of billions of samples. However, the gap here seems quite large.

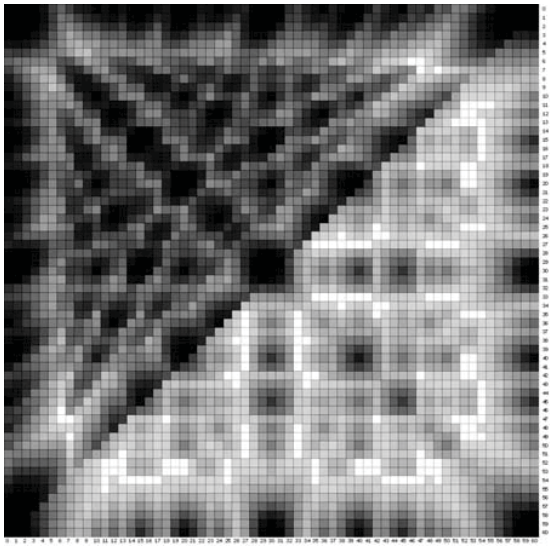


Figure 8: 37.1 billion samples of legal two-player BM6L configurations. Darker cells represent outcomes that occur more often; cells that are pure white never occurred and are likely to be unattainable. Note that the intensity represents the rank of occurrence, not the magnitude; in actuality, outcomes like $\langle 0, 0 \rangle$ occur *much* more often than others. 95.65% of the cells are filled.

Since we already know BM8L is universal, the remaining open problem is BM7L, whose results are in Figure 9. Indeed, after more than 30 billion samples the matrix is completely filled in; we have found an example configuration that achieves every outcome. Some of these were extremely rare, such as the solution for $\langle 11, 53 \rangle$ (Figure 10). In BM7L, two players can drink any amount.

6 The fractal geometry of $^{k}/_{n}$ Power-Hours

Note that all of the two-dimensional figures resemble one another even though they are fundamentally different (adding players, adding states, adding random trials). Even samples from BMML with three players (3D projected to 2D), which is shown in Figure 11, produces a similar pattern. This suggests that the combinatorial problem ("what outcomes are reachable from finite state machines that look kind of like this?") has some geometric structure.

Some of the patterns are easy to explain. The top-

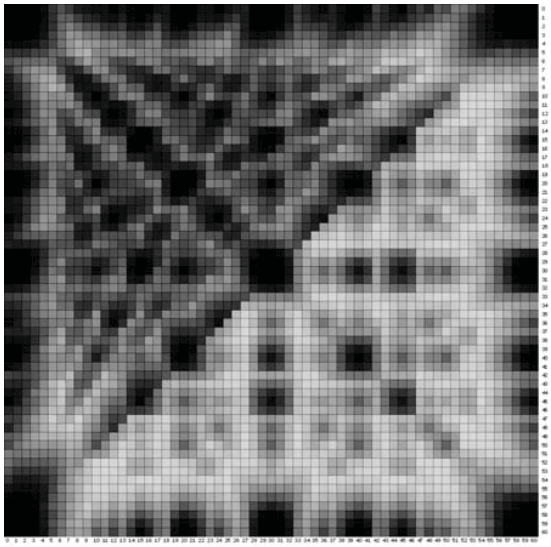


Figure 9: 30.7 billion samples of legal two-player BM7L configurations. Fewer configurations were sampled than in Figure 8 because they take somewhat longer than 6-state games to simulate, and a smaller proportion of random games are legal. Moreover, we stop after finding a solution for every cell, proving that BM7L is complete! The last cells found—an earlier version of this paper held these as open problems!—were permutations of $\langle 11, 53\rangle$ (Figure 10) and $\langle 49, 53\rangle$. Note that 53 drinks was also unattainable in a solo BM7L power-hour; this may in some sense be the "hardest" number of shots to drink in BM*s*L.

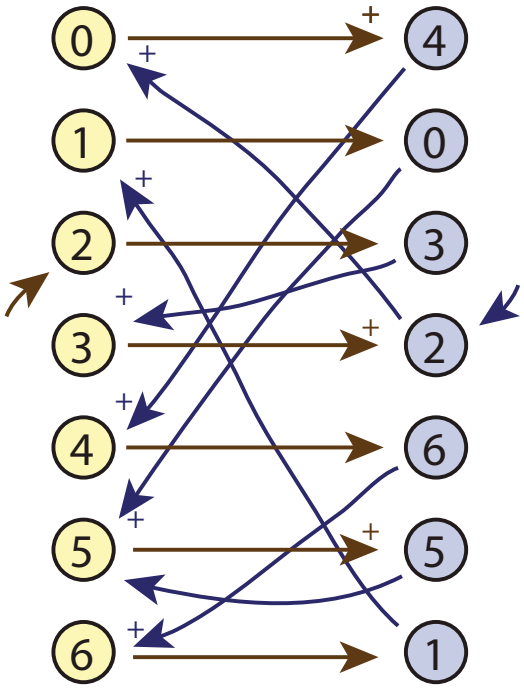


Figure 10: A solution for the elusive $\langle 11, 53\rangle$ two-player BM7L power hour! This one was found after 30.7 billion outcomes sampled, making it the most rare. By the end of the game, the two players are just passing back and forth two cups in state 5. A long lead-in beginning on the left-player's state 2 spans 12 different configurations (the 6 cup states for the two players) before entering the length-two cycle. Both cups travel along this lead-in, with one three steps ahead of the other. The right player drinks on every step until reaching the cycle (and then never again) for 11; the left player drinks during the cycle plus a little extra during the lead-in for 53. This configuration is quite flexible because the two players can make fine adjustments to their drink total by drinking or not drinking on the lead-in rules, which are executed just once or twice.

left half of the matrix is more populous than the bottom right, for example. This is because we can bound the total number of drinks by $60 \times c$, where c is the number of cups active in the game (same as the number of cups in the starting states). The top-left half is the region where this sum is less than or equal to 60; in two player games, both players must start with a cup in order to get an outcome in the bottom-right half. We also see distinct clumps around 0, 15, 30, 45, 60; these correspond to simple fractions ("drink every other time"; "drink three of four times") of 60. This is intuitive because the expressive power of BMML comes from the ability to form cycles of cup states and drink on some fraction of them. Clumps are formed around these values because of the possibility of preludes leading into the cycles (Figures 1, 10) that either drink (adding to the total) or don't (subtracting from it). Minor clumps form as echoes between the major ones, because a player may participate in two cycles of different length (Figure 4).

Of course, discretization effects compound and so the exact values of cells are not neatly predictable. Moreover, clumps interfere by overlapping; there are many different strategies for achieving $\langle 33, 33\rangle$. One way to

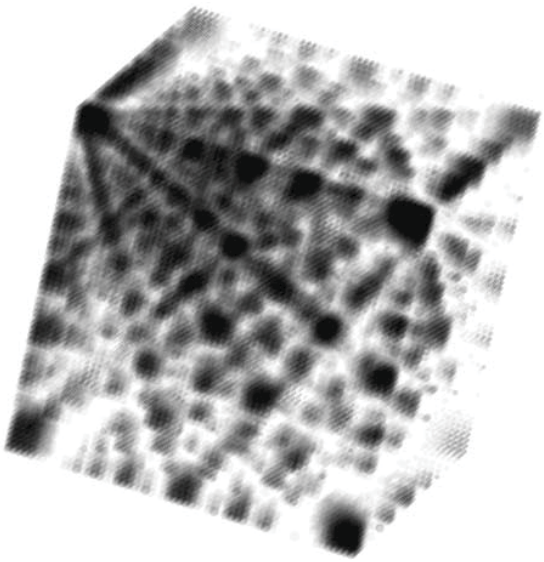


Figure 11: 490 million samples of three-player BMML Power Hours. The cube is rotated 15 ° along each axis, the top-left corner is $\langle 0, 0, 0 \rangle$, and the bottom right is $\langle 60, 60, 60 \rangle$.

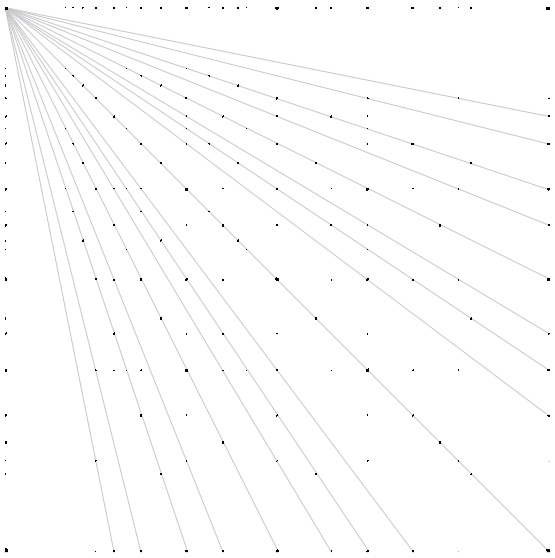


Figure 12: All possible outcomes for the first two players in 3-player power days. These are the same games as the 3-player power hours, but at this scale makes it clear the groupings and their sparsity in the limit. Lines plotted from $\langle 0, 0 \rangle$ to $\langle 60, k_2 \rangle$ and $\langle k_1, 60 \rangle$ show significant structure, but don't explain some of the interior points. These are probably games where a player participates in two cycles of different periods.

make the basic structure more visible is to extend the number of minutes that the game is played for. Figure 12 shows the utterly unhealthy three-player Power Day (BM3L). In it, the clumps become tiny dots, but some relationship among them along lines is clear. Interior points can probably be found as linear combinations of two of these lines; we exploit that exact structure in Section 4, in fact.

The less extreme $^k/_n$ Power-Three-Hours appears in Figure 13.

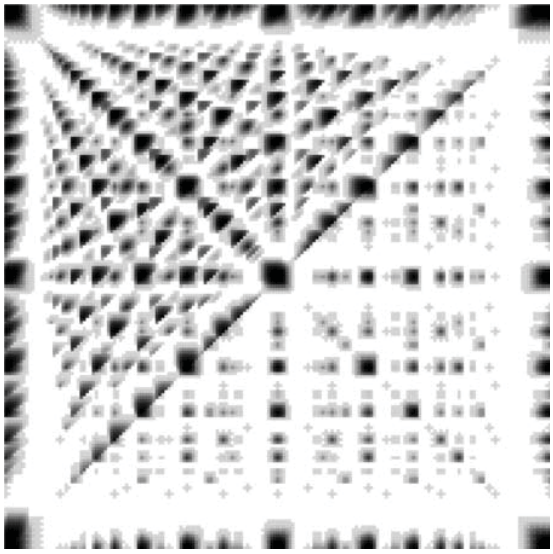


Figure 13: 6,456,764,116 samples of two-player BM7L Power Three-Hours.

7 Conclusion

This section summarizes the known bounds for BMsL, and states some conjectures, before concluding.

7.1 Known results

1. With one player, we have exact bounds on what is possible in the generalized case. With 8 cup states, a single player can drink 0–60 shots. Since each player can just play independently, this result extends to any number of players in BM8L. With

fewer than 8 cup states, not every k can be achieved alone.

2. With two players in BM3L or BM4L, it is not possible for one of the players to drink every k even if the other player helps her out.

3. With two players, we know that BM5L does allow one player to drink any k_1 if the other player assists. In fact we can achieve any $\langle k_1, k_2 \rangle$ where $k_2 \in \{0, 1\}$. No other k_2 can be used universally, though of course many other combinations are possible (Figure 7). Many pairs $\langle k_1, k_2 \rangle$ are known to be unattainable; this was established by exhaustively testing all possible configurations.

4. Open: Can BM6L achieve all $\langle k_1, k_2 \rangle$? Seems unlikely, given that random exploration plateaus with about 95.65% of the grid filled.

5. BM7L can achieve all $\langle k_1, k_2 \rangle$. This was established by sampling random games until we found an example for every $\langle k_1, k_2 \rangle$.

6. With three players in BM3L, one player can drink any number of shots if the other two players help.

7.2 Conjectures

Freedom: With two friends, you can drink any amount. We know that in a three-player game of BM3L, one of the three players can drink the k of her choice. This straightforwardly extends to $3 \times p$-player BM3L games. The **Freedom** conjecture is that with $p + 2$ players in BM3L, p of them may have their choice of $k_1 \ldots k_n$ drinks. If this conjecture fails, it probably fails for 4 players, which might have a feasible enumeration strategy.

Teetotaller: Someone can drink nothing. When $\langle k_1, \ldots, k_i, \ldots, k_p \rangle$ is achievable in BMsL, so is $\langle k_1, \ldots, 0, \ldots, k_p \rangle$. This conjecture would be trivial if not for the rule that requires us to drink the contents of a full cup if we want to pass it in a different state. This conjecture is true for all the graphics presented in this paper;[7] we can see that cells in the 0 column are always filled when some other cell in that row is filled. If this conjecture fails, it probably fails for BM1L or BM2L which have the least freedom per player.

In this paper I presented some new results in $^k/_n$ Power-Hour theory, as well as correct the historical record of some inebriated missteps. We saw that stochastic simulation, abstract interpration, and sampling are powerful tools for solving combinatorial drinking problems. We established some firm results for the classic game and some bounds for generalizations, as well as informally looked at some visualizations of its geometric structure. However, there are still several open problems in this field that demand further study.

[7] Actually, it is not established for (only) 11 minutes in Figure 13, but this is not a proper BMsL game as it takes place over 180 minutes. There should be solutions for 11, like in Figure 10; this is just a sample.

References

[1] Ben Blum, Chris Martens, William Lovas, and Tom Murphy, VII. Algorithms for k/n Power-Hours. *SIGBOVIK 2012*, pages 29–33, April 2012.

[2] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. *4th POPL*, pages 238–252.

[3] Tom Murphy, VII. The `textformat` library for Standard ML. `http://sourceforge.net/p/tom7misc/svn/HEAD/tree/trunk/sml-lib/textformat/`, 2013.

SIGBOVIK 2014 Paper Review

Paper 1: New Results in k/n Power-Hours

Robert Marsh, King Under the Mountain
Rating: 3 (weak accept)
Confidence: 2/4

Man, figure 5 does look really sweet. like whoa. you could take that and trim the edges off and use it as a boss in some kind of diagonal-scrolling space shoot-em-up. no really, that's blowing my mind. you could, like make an entire space invaders galaga thingy out of these charts. it would be rad as all get out.

On the other hand, one must consider the broader implications of research of this sort. In particular, should this conference encourage an attempt at a k/n power hour for any k, even a difficult-to-reach one like $\langle 11, 53 \rangle$? Let us first note that the finer things in life should be savored. However, there is such a thing as Beer 30 Light, and people are going to attempt such things whether we help them or not, so we might as well ensure they have the tools they need for slightly more responsible drinking.

In sum, this paper's significant contributions to cool fractally things seems to outweigh its potential harm to society.

Linear Logic Free

Carlo Angiuli

FREE

Description

TRY THE SMASH HIT LINEAR LOGIC FOR FREE!

Hundreds of logicians around the world are proving theorems in and about Linear Logic. Try it free on your paper or whiteboard device now!

★ ★ ★ ★ ★ ★ ★ ★

FEATURES

★ Resource Interpretation!
Innovative substructurality allows you to model resources as propositions! Menus, block worlds...the possibilities are endless!

★ Multiple Difficulties!
Prove theorems in intuitionistic linear logic, ⊗ for a challenge, try the included classical linear logic instead!

★ Prove With Friends & Adversaries!
Interactively challenge your adversaries to demonstrate that a theorem is true, without conveying additional information! Or collaborate with friends to prove new and exciting theorems—your choice!
(Collaboration only supported on whiteboard version.)

★ Hats!
Add some personal style to boring theorems with our new hat packs!

★ ★ ★ ★ ★ ★ ★ ★

If you like Linear Logic Free, consider upgrading to Linear Logic for some additional features:

★ Cut!
The premium-only cut rule allows you to save your progress at key checkpoints on your way to proving theorems!

★ !!
A whole new context of persistent propositions allows you to do ordinary logic and linear logic at the same time!

Like us on ScienceDirect: http://dx.doi.org/10.1016/0304-3975(87)90045-4

What's New

Hello, provers! In our latest update, we have added new consumable atomic propositions to Linear Logic. We have also moved the previously premium-only identity rule to the free version of Linear Logic, after complaints that grinding identity expansion was too tedious. Keep on proving, but remember to leave focus by taking frequent breaks!

In-App Purchases

- Hat Pack 01 ^ ~ -
- Hat Pack 02 

Developer Apps

- Linear Logic
- Proof Nets Free
- Proof Nets
- Chesslers of Catan

SIGBOVIK 2014 Paper Review

Paper 21: Linear Logic Free

Review #1
dabestcoq, gmail.com
Rating: 0 (strong reject)
Confidence: 4/4

this paper SUX i spent like $50 on a premise but wen i used it to prove an implicaton the premise was GONE WUT A RIPOFF!!!!!!! DO NOT USE THIS LOGIC!!!1! propazitonal logic is WAY better

Review #2
idbangthattype, gmail.com
Rating: 4 (strong acccept)
Confidence: 4/4

Reviewer 1 you idiot that's the point of linear logic. The paper is far from perfect. I for one wish the cut rule was included in the free version but I gave it a 4 to make up for all the dumbass reviewers.

Review #3
obamaisasocialist, gmail.com
Rating: 0 (strong reject)
Confidence: 4/4

ummmmmm reviewer 2 maybe we all dont want u buttin in. u no hu else told ppl wut 2 think? hitler

It still seems that black has hope in these extremely unfair variants of chess

Dr. Tom Murphy VII Ph.D. Ben Blum
Jim McCann, in italics ..... oh! .... nervous laughter .... no, that's not part of the name still. I stopped sa

1 April 2014

Abstract

CHECKMATE.

Introduction

Chess is an old-timey game that you already know. One problem with Chess is that it is hard; both players may struggle mightily in a game, expending their brain-sugars, and it is not clear who the winner will be. Another problem with Chess is that it isn't other games, and we're pretty much over it. In this paper we attempt to address both problems, with limited success. We show how to combine Chess with several other board games, in order to make it more predictable.

As usual for a Tom 7 SIGBOVIK joint, the results herein are real. The source code that was used to solve the games or prove that no winning strategy exists up to some depth can be found on the inter-net.[1]

1 Chesstego

Chesstego is a combination of the games Chess and Stratego. You already should know that every game in this paper is a combination of Chess and something, but I wanted to emphasize the portmanteau. All of the games will be named with portmanteau, and some of the names will be achingly bad.

In Stratego, each player begins the game by arranging his or her Stratego-pieces in a secret fashion on the board. In the world of Stratego, civilization is ruled by a leader known as Flag. The player's goal is to assassinate the opponent Flag, using a member of his or her army. However, which piece represents Flag is unknown!

There are many ways we might apply the Stratego system of governance to Chess.

Chesstego variation I. In this variation, each player decides in secret, before the game begins, which of his or her pieces is the actual King, that is, Flag. If this piece is checkmated, the game is lost. There are two subvariations: *I(a)*, where a player must announce *Check!* when Flag is under attack, and all of the normal rules about moving into (or castling through) check must be obeyed. In subvariation *I(b)*, which I prefer, the piece that is Flag may slip silently into and out of check, and the game only ends when that piece is captured.[2]

Chesstego variation I is a reasonable if slightly silly game, and it is difficult. It may even be more difficult than Chess due to the psychological mind-games that are possible. It can be hard to tell which player is winning, let alone which player will win. In this paper we are interested in variants of Chess that both *are other games* and *are predictable*. We'd like to give one of the players a clear strategy for winning.

Chesstego variation II. In this variation, each player decides in secret, before the game, which of her *opponent's* pieces is Flag. Same as before, if this piece is captured, the game ends instantly. But now, players don't even know which of their own pieces is their glorious ruler, Flag. This can be very exciting or titillating.

Unfortunately, there is no known winning strategy for White, and there are no winning strategies with fewer than 6 moves. This was proved by computer program. In fact, it was proved for a stronger case:

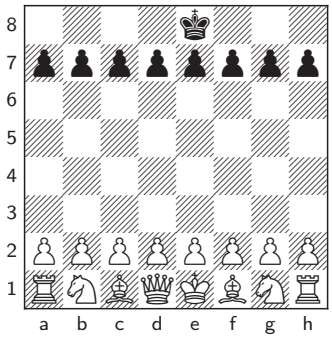
[1] In the Subversion repository at: `https://sourceforge.net/p/tom7misc/svn/HEAD/tree/trunk/chesstego/`

[2] In Chess proper, these formulations are nearly equivalent—except for rules like castling through check and some corner cases—but it is deemed important for movie drama that players be able to announce *Check!* and *Checkmate!* at times.

Chesstego variation III. In this variation, the first player, known as White, chooses in secret both the identity of Flag for his own populace, as well as the identity of Flag for his opponent. Even in this very unfair setup, Black can always survive for at least 6 moves.

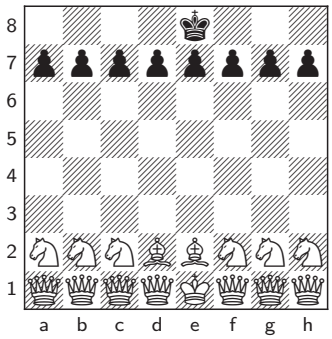
This will not do! Black just has so many options with all those pieces. Perhaps if he were handicapped somewhat?

Chesstego variation IV. In this variation, White chooses the identity of both Flags again, in secret without telling her opponent, and also Black does not get any good pieces, just pawns. Like this:



Lo! This version is finally satisfactory. In this version we need not equivocate over who shall win. White picks one of her very safe pieces (e.g., one of the rooks) as Flag, and chooses Black's b7 pawn as Black Flag. White's winning move is c2c4 and then Qb3, with Black Flag unable to escape the B file (Figure 1). Choosing the f7 pawn works as well.

Chesstego variation V. This variation is just like *IV* but White gets super good pieces everywhere instead of having some dumb ones, and Black still gets bupkiss:



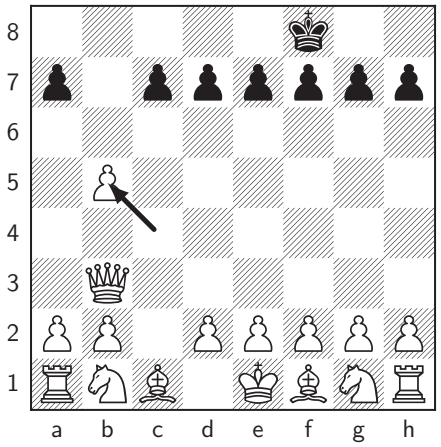


Figure 1: Example hopeless match: 1. c4 Kf8 2. Qb3 b5 3. xb5 1–0

Now there are many more winning strategies, but the one from *IV* works as well.

Other variations. Incidentally, my on-line version of Chess called SICO[3]—where you just play a single move in a random game—has for about six years had variations called "center wall" and "barricades"[1] which were inspired by the layout of the Stratego board. However, these cannot be considered worthy of portmanteau, as the rules are basically just Chess rules.

2 Cluess

This game, known as *Chuessdo* in the United Kingdom, is a cross between Chess and *Clue*. In this version, the player called White is known as Mrs. White, and the player called Black is known as Professor Plum.

3 Chess Who?

The board game *Guess Who?* disappointed millions of children with its delightful television commercial that far outstripped the capabilities of the actual game. This required them to add a disclaimer to the commercial, "Game cards do not actually talk." In *Guess Who?*, players take turns trying to guess the identity of the opponent, among a fixed set of personalities with traits like curly hair, straight hair, brown hair, short hair,

[3]On the internet at: `http://snoot.org/toys/sico/`

bangs, long hair, blonde hair, red hair, or glasses. One player asks, "Do you have curly hair?" and the other says "Yes" or "No" and then the first player can eliminate all the remaining personalities that don't have the trait. It's basically binary search, but for kids.

In *Chess Who?*, which is the chess version, players have two different options on each turn. They can either move a piece like normal, or ask a question of their opponent. The question must be about the physical characteristics of the pieces, for example, "Does your piece have curly hair?" No pieces have curly hair, so nothing happens. The opponent might ask a different question, on the next turn, like "Does your piece have straight hair?" No pieces have straight hair, so nothing happens. On the next turn, suppose the player asks, "Does your piece have brown hair?" No pieces have brown hair, nor any hair at all, and no pieces have any brown parts at all either. Stop asking. Next, Black asks, "Does your piece have crenellations?" Both of White's rooks have crenellations, so they are eliminated from the board (Figure 2).

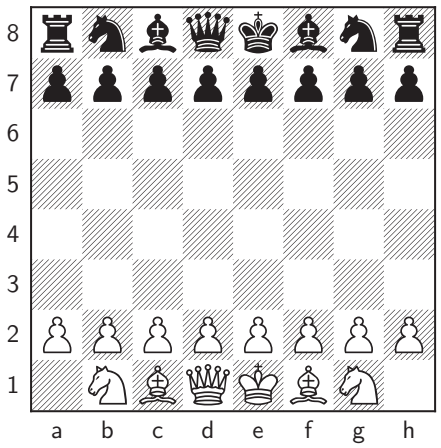


Figure 2: The board after 1. Does your piece have curly hair? ... Does your piece have straight hair? 2. Does your piece have brown hair? ... Does your piece have crenellations?

Next, White might ask, "Does your piece have a cross on his hat?" This matches both Bishops and the King, but Black responds "Check!" because it is illegal to ask a question that would eliminate the King.

This variation produces draws easily. An example Nash Equilibrium: White begins by asking, "Does your piece have three lobes like a truncated snowman, *or* crenellations, *or* horse ears, *or* two weird long shoulder-ribbons that I don't know what they are, *or* five pointy tines on her crown?" This matches all of Black's pieces except his king, so he removes them all. But then Black asks: "Does your piece *not* have a lumpy crown?" which matches all of White's pieces except his king. So all we have left is the two kings, which is known to be a draw [2].

4 Chessy Crush Saga

Chessy Crush Saga is a hybrid of Chess and the popular and lucrative phone-game *Candy Crush Saga*. In this game, the regular moves of chess are allowed, but it is also possible to *crush* chess pieces into candies. *Chessy Crush v1* follows the rules of *Candy Crush* closely: When a piece moves such that three pieces of the same color are in a row (either horizontally or vertically), those pieces are all removed. Usually this is a tactical misstep in Chess, as three of one's own pieces are destroyed for nothing. However, as in *Candy Crush*, rewards are given when more than three like pieces are destroyed at once. The simplest is to award the player with a Candy Rook at the place where the moved piece ended up. A Candy Rook moves like a rook, but if it is crushed, it destroys all of the opponent's pieces on the file (column) it currently occupies.[4]

As another simplification, we remove all of the proper pieces from the black files, except for the King, in honor of the software company that makes *Candy Crush*, "King." A densely packed board with pieces that can move backward is dangerous, for it is easy to destroy one's own king by moving out of and then back into the back file, crushing the whole thing (Figure 3).

With just pawns, it seems that the best way to gain advantage for White is to crush four pawns to create a Candy Rook before Black does, then menace Black with it. Unfortunately, moving four pawns into formation before black can interfere is not possible; Black can react to White's first move and advance one of his own pawns towards the construction (Figure 4). Moreover, if he does, and trades for one of White's pawns (or better, the resulting candy), then Black is probably in a superior position, White having crapified his pawn structure in pursuit of candy. Worse still, crushing a successful Candy Rook is not that menacing, since Black can sim-

[4]In formal *Candy Crush*, this is the vertical striped candy, which is awarded when the piece was moved vertically. To keep this variant simple, we always award a "vertically striped" Candy Rook, since it seems with only pawns, only vertical moves are likely. (This later turns out to be a bad assumption.)

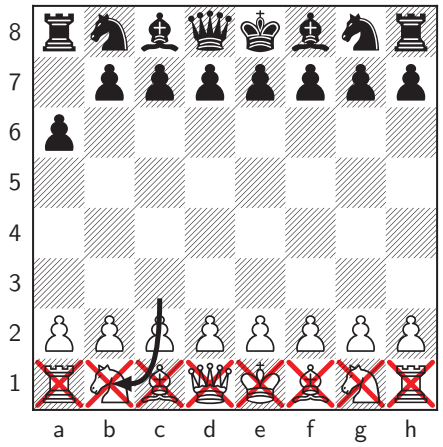


Figure 3: "Fool's Mate" in the rejected *Candy Crush v0*, played with normal pieces in the back ranks. White self-mates in two moves, crushing all his own pieces *e.g.* with: 1. Nc3 a6 2. Nb1 (??) 0–1

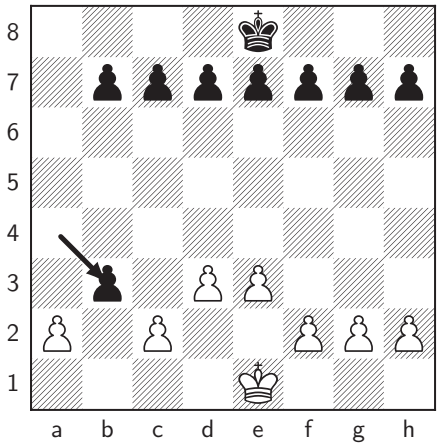


Figure 4: Black interfering with White's Omega Weapon main line: b3, d3, e3, c3 *(making candy)*, Rc2, Re2++. Black can progress pawns fast enough to capture White's. She can also move her King, making the candied rook less devastating. Ultimately, White's one-tempo advantage is not obviously winning.

ply move his king out of the way.

To correct these problems with balance (*i.e.*, that the game may be balanced), we simplify the candying rules: Any crush produces a Candy Rook, including ones of length 3. This allows White to create a Candy Rook before Blacks pawns can reach her. Furthermore, we stipulate that the kings cannot move (they don't have legs anyway, this is plain to see, so let's be realistic). In *v2*, the Candy Rook is extremely powerful, since not only does it destroy an entire file (the e file being the likely target, since it contains the immovable King), but when it is used (by crushing) it is also replaced, since a crush always yields candy. Therefore, White's most straightforward strategy is to quickly create a Candy Rook, then crush it in the e file, which wins. This should be a line where White's tempo advantage makes it mostly immune from interference, since even a check is answered by destroying the opponent's king.

The idea behind White's Omega Weapon is to create a triplet of pawns on rank 3, either c3–e3 or e3–g3. White begins by moving her king's pawn to e3. Black can interfere with one side of the triplet (Figure 5), but only one. After 1. e3, if . . . f5 or h5, then White will continue creating the triplet in c3–e3. Otherwise, she is safe to create it in e3–g3. Once the Omega Weapon is constructed, she need only move it to e2 (which is free due to the first move, and will be adjacent to at least two pawns). This destroys the Black king, which cannot have moved, due to the rules. Unfortunately, this strategy can be foiled by Black with a clever[5] mutual suicide, as shown in Figure 6. This strategy is not easily defended against, nor is it straightforward to fix the rules of the game;[6] so it seems *Chessy Crush Saga v2* also retains hope for Black, even though the odds initially seemed stacked against her.

5 Future work and conclusions

Many other games can be combined with Chess to ruin Chess. For example, consider *Chess Tac Toe*, *Chesslers of Catan*, *Chesstris*, and *Hungry Hungry Hipposchess*. Combining Chess and Sokoban is impossible, obviously. The combination of Chess and Battleship is *Battlechess*, published in 1988 by Interplay.

[5] This strategy was discovered by computer, like most clever things these days.

[6] It seems perhaps we can eliminate Black's draw strategy by restoring the idea of horizontally striped Candy Rooks (see footnote above). I implemented this. Unfortunately, although the move that ends the game in a draw is a horizontal move, the move that creates the Candy Rook is a vertical one. Therefore, Black's Candy Rook is vertically striped, and can successfully destroy the White king (though it is then replaced with a horizontally striped Candy Rook to observe the empty thrones).

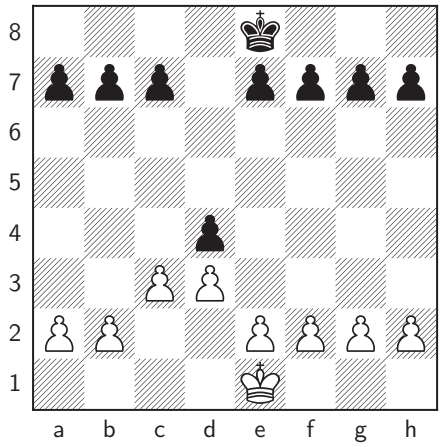


Figure 5: White may not use a fixed strategy to construct her Omega Weapon. The position shown is after 1. c3 d5 2. d3 d4. If 3. e3 *(making candy)* then . . . xe3, obviously. White can make candy with 3. b3, but now e2 is not clear to crush the Candy Rook here and destroy the Black king.

In conclusion, San Dimas High School Footchess rules!

References

[1] Tom Murphy, VII. SICO chess variations, April 2008. `http://snoot.org/toys/sico/variations.html`.

[2] John Nunn. *Secrets of pawnless endings*. Gambit publications, 2nd edition, May 2002.

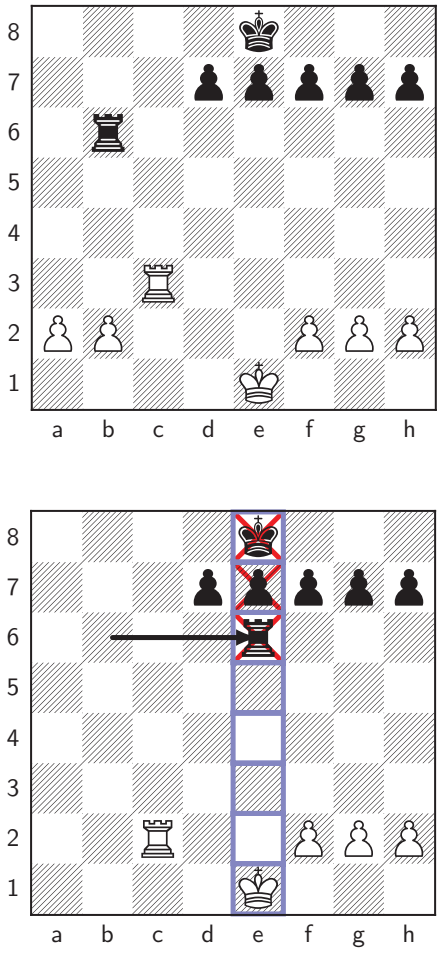


Figure 6: Black's surprising draw strategy in *Chessy Crush v2*. White begins her Omega Weapon: 1. e3 c6 2. d3 a6 3. c3 *(making candy)* b6 *(making candy)* we have the board in (a). It seems that Black is stuck after 4. Rc2 *(making candy)*, since 5. Re2 *(making candy)* will crush White's candy rook and destroy the entire e file, including Black's king. Black defending with 4. . . . Re6 initially seems pointless (White will destroy the rook as well so the check is irrelevant—this indeed refutes 4. . . . Rb1) but Re6 actually creates a *vertical* three-piece crush, sacrificing Black's king to destroy the e file first, which also destroys White's king! Such mutual destruction is not possible in traditional chess, but it seems appropriate to consider it a draw in Chessy Crush Saga v2. It appears that White can head off this line with 4. Rc8++, a totally vanilla back-rank mate, but this is not actually mate with the same 4. . . . Re6 *(making candy)* escaping to a draw.

Track 88

Concerning Local Affairs

1. **PARTIAL TRANSCRIPT OF DEAN'S SPEECH TO THE FACULTY OF THE MELLON COLLEGE OF SCIENCES**

   Clint Falamenos

   Keywords: news, lane center, college of sciences

2. **Analysis of the Effects of Substantial Lane Closure During Afternoon Peak Along a Heavily-Traveled Urban Arterial Choke-Point**

   Nicholas Fudala

   Keywords: knowledge, pagers, RATPAG

3. **Yet Another Application of Our Pet Technique**

   Chester Francis and Nicholas Fudala

   Keywords: complexity, multiple patents pending, nobel peace prize

4. **Please Don't Let Open House Destroy the Universe**

   Jenn Landefeld

   Keywords: open house, destruction of universe, herding cats

TRANSCRIPT OF DEAN'S SPEECH TO THE FACULTY OF THE MELLON COLLEGE OF SCIENCES -- March 18, 2014

Dear friends, we have gathered here today in connection with an issue that is of vital, historic significance to all of us. A referendum was held among the faculty of the Ray and Stephanie Lane Center for Computational Biology on March 16 in full compliance with university procedures and academic norms.

More than 82 percent of the faculty took part in the vote. Over 96 percent of them spoke out in favour of reuniting with the Mellon College of Sciences. These numbers speak for themselves.

To understand the reason behind such a choice it is enough to know the history of the Lane Center and what the College of Sciences and the Lane Center have always meant for each other.

Everything in the Lane Center for Computational Biology speaks of our shared history and pride. In 1989, the first degrees were awarded in the undergraduate computational biology program at Carnegie Mellon. Courses developed for this program stimulated interest among graduate students, as well.

In 1999, Mellon College of Science received a grant from the Merck Company Foundation to create a new program in computational biology and chemistry, which supported both undergraduate and graduate students, thereby helping to stimulate development of interdisciplinary, collaborative projects. A major limitation of the Merck program was that students had to be enrolled in one of the traditional Ph.D. programs. Five years later, in 2004, a new Ph.D. program in computational biology, the Joint Carnegie Mellon University - University of Pittsburgh Ph.D. Program in Computational Biology (CPCB), was established, with the first students enrolled in Fall 2005.

In 2007 The Ray and Stephanie Lane Center for Computational Biology is created as the culmination of many years of development and recruiting efforts, and with the goal of realizing the potential of machine learning for expanding understanding of complex biological systems. A primary focus of the center is developing the computational tools that will enable automated creation of detailed, predictive models of biological processes, including automated experiment design and data acquisition.

In 2009, for a number of reasons – may God judge them – the Lane Center became its own academic department and transfered to control of the School of Computer Science. This was done with no consideration for the academic make-up of the faculty. Whether this decision was backed by a desire to win the support of the tenured Computer Science faculty establishment, or to atone for the prior lack of a proper PhD program, is for historians to figure out.

What matters now is that this decision was made in clear violation of the university norms that

were in place even then. The decision was made behind the scenes. Naturally with the disconnected leadership of the times, nobody bothered to ask the faculty of the Lane Center. They were faced with the fact. Dozens of faculty went to bed in one department and awoke in another, overnight becoming academic minorities.

Now, many years later, I heard faculty of the Lane Center say that back in 2009 they were handed over like a plasmid. This is hard to disagree with. And what about the Mellon College of Sciences? It humbly accepted the situation. This college was going through such hard times then that realistically it was incapable of protecting its interests. However, the faculty could not reconcile themselves to this outrageous historical injustice. All these years, citizens and many public figures came back to this issue, saying that the Lane Center is historically a science department. Yes, we all knew this in our hearts and minds, but we had to proceed from the existing reality and build our good-neighbourly relations with the School of Computer Science. What we proceeded from back then was that good relations with SCS matter most for us and they should not fall hostage to deadlock departmental disputes. However, we expected SCS to remain our good neighbour, we hoped that faculty in the College of Sciences and biologists in SCS, especially the Lane Center, would work in a friendly, collaborative and inter-disciplinary environment.

However, this is not how the situation developed. Time and time again attempts were made to deprive biologists of their disciplinary traditions, even of their technologies and to subject them to forced assimilation. Recently, the so-called authorities of SCS even drafted a committee resolution to revise the programming language policy, which was a direct infringement on the rights of disciplinary minorities.

Those who opposed this treatment were immediately threatened with repression. Naturally, the first in line here was the Lane Center, the biologically-oriented Lane Center. In view of this, the faculty of the that department turned to the Mellon College of Sciences for help in defending their rights and livelihoods. Naturally, we could not leave this plea unheeded; we could not abandon the Lane Center and its members in distress. This would have been betrayal on our part.

To this end, we had to help create conditions so that the Lane Center for the first time in history were able to peacefully express their free will regarding their own future. However, what do we hear from our colleagues in the School of Drama? They say we are violating the norms of university policy. Firstly, it’s a good thing that they at least remember that there exists such a thing as university policy – better late than never.

Moreover, the Lane Center authorities referred to the well-known School of Architecture precedent – a precedent our Dramatic colleagues created with their own hands in a very similar situation, when they agreed that the unilateral separation of the School of Architecture, exactly what the Lane Center is doing now, was legitimate and did not require any permission from the school's central authorities.

I do not like to resort to quotes, but in this case, I cannot help it. Here is a quote from another official document: the Written Statement of the School of Drama. I quote: “Declarations of independence may, and often do, violate collegiate policy. However, this does not make them violations of university policy.” End of quote. They wrote this, disseminated it all over the university, had everyone agree and now they are outraged. Over what? The actions of Lane Center faculty completely fit in with these instructions, as it were.

Like a mirror, the situation in the Lane Center reflects what is going on and what has been happening in the university over the past several decades. Key university institutions are not getting any stronger; on the contrary, in many cases, they are sadly degrading. Our Dramatic partners, prefer not to be guided by university guidelines in their practical policies, but by the rule of force. They have come to believe in their exclusivity and exceptionalism, that they can decide the destinies of the university, that only they can ever be right.

We understand what is happening; we understand that these actions were aimed against the School of Computer Science and the College of Sciences. And all this while we strived to engage in dialogue with our colleagues in Drama. We are constantly proposing cooperation on all key issues; we want to strengthen our level of trust and for our relations to be equal, open and fair. But we saw no reciprocal steps.

[To access the rest of this transcript, please listen to this message from our sponsors...]

Analysis of the Effects of Substantial Lane Closure During Afternoon Peak Along a Heavily-Traveled Urban Arterial Choke-Point

Nicholas Fudala

February 24, 2014

1 Introduction

We have been advised to begin by clarifying that, yes, we are the traffic engineers responsible for the study at the heart of "Bridgegate" and that Governor Chris Christie "had no knowledge of this—of the planning, the execution, or anything about it."[1] With that out of the way, we'd like to thank the media for the unexpected interest in our work. While we once struggled to receive approval or even acknowledgment for the legitimacy of our research interests, that being our queue length prediction system, we now find ourselves overwhelmed with inquiries seeking the answer to "what exactly is this traffic study all about?"

2 Background

Traffic congestion is an unavoidable fact of modern life. Sometimes you approach a line of red tail lights and have no idea how long it stretches ahead of you. So one day, in 1986, while stuck in such a predicament, we devised such a method to calculate queue length and report its findings to each subscriber's one-way pager. Though offering no congestion-free alternatives, the system would offer peace of mind in knowing exactly how long the subscriber would remain stuck in the queue, away from friends and family, after a long day at work. Nineteen years later, while still working out the bugs, an upstart search engine named Google released their "Google Maps" and its traffic observation feature, effectively duplicating our effort for non-pager devices. Therefore, we seek to provide an alternative to Google and subsequent traffic observation websites, apps, and local news broadcasts.

3 Methodology

We were approached one day, to our complete surprise, by the Transportation Commissioner of the State of New Jersey to conduct a traffic study on the George Washington Bridge, connecting Fort Lee, NJ and New York City. Inspired by the original cause of the congestion that would spark the idea for our queue length prediction system we decided to simulate a mattress on the freeway situation by closing two of the three travel lanes from the bridge to Fort Lee. Our proprietary algorithm relied on public-

use roadway sensors and cameras and converted all necessary data into a text message provided to subscribers (which, of course, there were none in this beta testing stage). Information was instead unintentionally encrypted and placed in a password-protected file.

4 Results and Future Research

Research is still ongoing as the authors are continually attempting to guess the password protecting all relevant data and information from the one-day traffic study. We are confident, as evidenced by the unexpected outpouring of media and worldwide interest, that we will find support in further study of our system. In fact, Governor Christie has been quoted saying "Go ahead."[2] We believe it necessary to investigate worst-case scenario mattress-on-freeway situations and propose research on complete lane closure during afternoon peak as well as travel access restricted to one 8-foot shoulder during an ice storm. We appreciate the continued support of the Governor, who agrees that "we need to fix this problem, because I'm the boss."[3]

5 Acknowledgments

Special thanks to the Transportation Commissioner of the State of New Jersey, the Port Authority of New York and New Jersey, and the Governor's Office of New Jersey for providing unprecedented access with virtually no notice to such a vital bottleneck.

References

[1] Kate Zernike. Christie linked to knowledge of shut lanes. http://www.nytimes.com/2014/02/01/nyregion/christie-bridge.html, January 31, 2014.

[2] The Washington Post. Full transcript: N.J. Gov. Chris Christies Jan. 9 news conference on George Washington Bridge scandal. http://www.washingtonpost.com/politics/transcript-chris-christies-news-conference-on-george-washington-bridge-scandal/2014/01/09/d0f4711c-7944-11e3-8963-b4b654bcc9b2_story.html, January 9, 2014.

[3] CNN Political Tracker. Chris Christie: 'If I was in the Senate right now, I'd kill myself'. http://politicalticker.blogs.cnn.com/2013/10/11/chris-christie-if-i-was-in-the-senate-right-now-id-kill-myself/, October 11, 2013. CNN Political Tracker.

Yet Another Application of Our Pet Technique

Chester Francis Nicholas Fudala

February 25, 2014

Abstract

Traffic lights are the worst. In most cities, their timing is probably just random guessing by some hillbilly local who can't count past ten, and we're unaware of more advanced techniques in the literature. We apply the extensively tested and occasionally accepted Francis-Fudala technique to the optimization of a simple two-phase isolated traffic signal, and discuss some selected benefits.

Figure 1: A traffic light.

1 Introduction

Traffic lights, also referred to as traffic signals, traffic lamps, signal lights, stop lights and robots [1], are defined as "a road signal for directing vehicular traffic by means of colored lights" [2]. See Figure 1.

They were first introduced in 1868 in London [1]. When you wait at a red light, you waste time, and your car wastes gas, so it's bad for both you and the environment. To our knowledge, there is no known way to improve upon random guessing for green time distribution, either in the literature or in practice [1, p. 1].

2 Our Approach

A potentially wildly successful and robust technique for optimal decision-making is the Francis-Fudala technique [3, 4]. A non-linear, multi-objective, Bayesian procedure utilizing dynamic programming, neural networks, and various data mining approaches comprising several of the leading optimization techniques in sequence, and occasionally produces results superior to when used individually.

The algorithm uses a multi-step process to output a single value of 1 or 0 (change or hold current phase) based on an input of vehicles on each link, geometric mean of vehicle locations, median of time-to-intersection, time since last phase change, current time in MATLAB (nanoseconds since January 0, 0), friction of road surface (in simulation assumed to be 1), vehicle (RGB), mean engine displacement (cc), min-

imum recommended tire inflation (psi), and various barometer readings.

Inputs are first computed using a symbolic regression tool very similar to, but definitely not, Cornell University's Eureqa [5], for a minimum of five days of simulation on a Pentium 4 Quad Core. These outputs are then analyzed with a stochastic artificial neural network that is not unlike, but also unlike, the one available in MATLAB's neural network package. The third step is a proprietary combination of Google Translate German to English (but not) and WinZip.

In case you can't tell, this is really complicated and therefore better.

The last digit of the WinZip file size in bytes is utilized in the next step, unless that digit is neither zero nor one, in which case the technique is repeated until an acceptable value is outputted. The boolean output undergoes the final step, one that is so shocking and brilliant that articulating it in this paper could destabilize all of science (for an even more cryptic explanation, see [6]). Rest assured it is so amazing and brilliant that it completely guarantees success and is in no way possible to improve or even understand, as evidenced by the US Patent Office refusing to even acknowledge our many applications.

2.1 Past Performance

The Francis-Fudala technique has been applied with various degrees of success and non-success to a wide array of problems in the fields of botany [7], psychology [8], paranormal psychology [9], stock-picking [10], lock-picking [11], off-track betting [12], on-track betting [13], guilt predictions on episodes of Dateline NBC [14], automated tweet scheduling [15], alcohol content per unit currency [16], number of M&M's in a cylindrical container [17], the Aanderaa-Karp-Rosenberg conjecture [18], and number of Skittles in a cubical container [19]. The algorithm has routinely placed in the top 80% of submissions in multiple data mining competitions hosted on Kaggle.com, and has been nominated multiple times for the Nobel Peace Prize (the Prize in Economic Sciences surprisingly restricts self-nomination).

2.2 Traffic Signal Application

Our technique was applied to a common two-phase traffic signal at a theoretical intersection. Modeling driver behavior proved particularly difficult, due to the complete lack of vehicle dynamic models found in an extensive search of the literature that were either free, in front of paywalls, or without lots of math. In this application, the intersection was modeled in NetLogo [20], assuming cubic vehicles with point-mass properties, accelerations of [inf, -inf], using the car-following model employed in Frogger™. A visualization of our network is shown in Figure 2.

Figure 2: Traffic signal application.

2.3 Results

Analysis was abbreviated due to the excessively long computational requirements of our algorithm. Each second of simulation required between 4 and 6 hours of computational time, so that a simulation of one hour of evening peak period traffic required 33 days of computation. Significance was therefore not calculated due to the low sample size of $n = 3$. Results are shown in Table 1.

Technique	Delay (s/veh)	Stops
Random Guessing	56.7	421
Naïve	54.5	450
Francis-Fudala	**54.2!!!**	3452

Table 1: Performance of the Francis-Fudala Technique compared with benchmarks. Note the improved performance in delay. It's in the middle bottom of the table.

3 Future Research

Although our testing conclusively demonstrated the vast superiority of the Francis-Fudala technique, there remain a few areas for further research. Potential topics include the performance of the Francis-Fudala technique on a three-phase intersection, four-phase intersection, five-phase intersection, six-phase intersection, seven-phase intersection, eight-phase intersection, nine-phase intersection, ten-phase intersection, eleven-phase intersection, twelve-phase intersection, two-legged signalized roundabout, three-legged signalized roundabout, four-legged signalized roundabout, five-legged signalized roundabout, six-legged signalized roundabout, seven-legged signalized roundabout, short-ramp meters, long-ramp meters, on-ramp meters, off-ramp meters, interchange ramp meters, diamond interchanges, diverging diamond interchanges, and theorized autonomous vehicle intersection with individualized lights for each vehicle. Testing on each of these signal configurations would be incomplete without further testing of the effects of standing snow, falling snow, falling snow blowing sideways, unexpected snow, expected snow, ice, rain, icy-rain or sleet, hail, wind, glare, earthquakes, solar eclipses, lunar eclipses, proportion of trucks, proportion of bicyclists, presence of pedestrians, presence of particularly distracting pedestrians, sensor failures, police sobriety checkpoints, recurring congestion, non-recurring congestion, and random traffic light bulb burnouts. Obviously each of these combinations requires its own paper, and probably book.

References

[1] Wikipedia. Traffic light. `http://en.wikipedia.org/wiki/Traffic_light`.

[2] The Free Dictionary by Farlex. Traffic light. `http://www.thefreedictionary.com/traffic+light`.

[3] Chester Francis. *Artificial Complexity*. PhD thesis, University of Burgundy, submitted.

[4] Nicholas Fudala. *Universal Complexity*. PhD thesis, University of Burgundy, submitted.

[5] Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *Science*, 234(5923):81–85, April 2009.

[6] Chester Francis and Nicholas Fudala. *An Even Newer Kind of Science*. Publisher: Author, 1st edition, January 2003.

[7] Chester Francis and Nicholas Fudala. Towards predictive coloration of the Eastern Maple. *Journal of Botany Enthusiasts*, submitted.

[8] Chester Francis and Nicholas Fudala. Approaching an understanding of the bias effect in handicapping of equine racing. *Journal of Sports Psychology*, submitted.

[9] Chester Francis and Nicholas Fudala. Approaching an understanding of the bias effect in handicapping of equine racing in the presence of ghosts. *Journal of Paranormal Psychology*, submitted.

[10] Chester Francis and Nicholas Fudala. En route to PENNY STOCKS ARE THE IPHONE OF INVESTING! *The Motely Fool*, submitted.

[11] Chester Francis and Nicholas Fudala. In the vicinity of a predictive approach to fastening device liberation: a quasi-philosophical perspective. *Journal of Speculative Research*, submitted.

[12] Chester Francis and Nicholas Fudala. Approaching an understanding of the bias effect in handicapping of equine racing. *Daily Racing Form*, submitted.

[13] Chester Francis and Nicholas Fudala. A novel stereoscopic camera object identification test case: classification of discarded ticket status at Yonkers Raceway. *IEEE Transactions on Entrepreneurship*, submitted.

[14] Chester Francis and Nicholas Fudala. It's often the married man. *Journal of Media Studies*, submitted.

[15] Chester Francis and Nicholas Fudala. Hashtag: profit. *Harvard Business Review*, submitted.

[16] Chester Francis and Nicholas Fudala. Alcohol dependence and poverty, but not the way you think. *Journal of Substance Abuse*, submitted.

[17] Chester Francis and Nicholas Fudala. Step right up! How to beat a carnie at his own game, and impress your friends. *Topics in Social Sciences: Special Issue on Carnival Culture*, submitted.

[18] Chester Francis and Nicholas Fudala. Towards preliminary understanding of the significance, magnitude, and difficulty of the Aanderaa-Karp-Rosenberg conjecture: initial thoughts. *Nature*, submitted.

[19] Chester Francis and Nicholas Fudala. Step right up: Particle-count prediction in a linear, equaliteral non-spherical environment. *Proceedings of the National Academy of Sciences*, submitted.

[20] Uri Wilensky. Netlogo. `http://ccl.northwestern.edu/netlogo/`.

SIGBOVIK 2014 Paper Review

Paper 5: Yet Another Application of Our Pet Techn

Robert Marsh, King Under the Mountain
Rating: 1 (strong reject)
Confidence: absolute/4

There are several problems with this paper. The modeling of vehicles is rather poor - at the very least their acceleration in the road's reference frame is limited by relativistic effects. Furthermore, modern vehicles are curvy, not dumb and boxy (except the Nissan Cube, which isdumb, boxy, and curvy, truly a marvel of modern engineering) and thus would be best modeled as spheres. In addition, the alternatives they test against don't include the current industry standard dime-an-hour hick, suggesting that they were cherrypicked to appear more favorable. And besides, they didn't cite *anything* I did, so I don't see why I should let them get published.

Please Don't Let Open House Destroy the Universe

Jenn Landefeld*

Carnegie Mellon University
jennsbl@cs.cmu.edu

Abstract

Exploration of the multi-dimensional space-time rift potentials [1] created through appointment scheduling containing m admitted students crossed with n faculty crossed with y current graduate students willing to speak with each other within two eight-hour sets of 30 minute meeting slots. The two day meeting period is also sprinkled with a series of events that interfere with multiple potential 30 minute meeting times causing admitted students, current students and faculty to not be able to meet with those they preferred to meet with. Events also allow chance meetings which will generate emails requesting additional meetings or changes to schedules that will cause ripple effects across the entire appointment structure.

I. Introduction

In this paper we explore the relatively undocumented practice of inviting multiple students admitted to a Computer Science Ph.D. Program to descend upon a given university to attempt to glean information from faculty and current Ph.D. students in an effort to determine whether said university and its faculty are a suitable fit for their research interests. Through this exploration we will determine if there is a real potential for the creation of a multi-dimensional space-time rift that could be catastrophic in proportion.

II. Methods

For examining this dilemma we will assume the following defined variables are in play:

Table 1: *Number of People Needing Appointments*

Faculty	Admit	Current Ph.D.
n =105	m =59	y =49

To accomplish the appointment schedule for matching m (where m is those who are able to travel to visit the given University at the set schedule time) across n and y we need to optimize the schedules to get as many m into the offices of n they requested to meet while also optimizing the schedules to accommodate the availability provided by n, cross-referenced with their level of willingness to actually meet with a given m.

When this optimization[2] fails, a second round attempt will be made to provide any faculty interaction in a given area of research before cross-checking the list of current Ph.D. student availability to match admitted students to with current students whose advisor might potentially be one of the faculty the admitted student requested to meet with.

Faculty and current students are contacted in advance of the student visits to assess their availability for 30 minute meeting slots during morning and afternoon daylight work hours on two standard university workdays comprising a total of 5-6.5 hours of meeting slots each day.

*Thanks to Rob Simmons for a much more concise title than this paper began with. Articles with short titles are cited more often.[3] Thanks also to Martha Clarke for years of dedication to making this process work for the Computer Science Department and without whose retirement I wouldn't have had the privilege of taking on the daunting task explored in this paper. There are so many more I could thank, but I don't want to make them feel uncomfortable by pointing out their contributions to the added complexity of the dilemma.

Appealing to the *"What's in it for me factor"*

- Get me students to spend funding on – so funding isn't taken back by (a) department, (b) school, (c) university, (d) benefactor (usually government funding with spending timeline/strings attached)
- Get me students to advance the research (more bodies on the problem)
- Get me students so I can write more grants to get more funding to fund the students I'm recruiting.

Appealing to the *Rock Star* factor

- You can have you name on even more submitted papers?
- If you are tired of traveling and talking to people about the research you can send a student.
- Get more students here to work with other faculty to keep *those faculty* busy and to stop bugging you to do things.

Appealing to the *"Okay there's food I guess I will attend that"* factor

- We all know faculty and students in computer science departments like to eat - provide food!

Research we didn't have time to fully define or apply findings from[1]

- Managing the Absent Minded Professor
- Managing the Rock Star factor
- Mitigating the Curmudgeon factor
- Mitigating the Procrastinator
- Managing unrealistic expectations of who will get to speak to whom (this can be admit or faculty based)

III. Conclusions and Future Research

Obvious conclusions (Duh)

- Admitted students are perfectly happy to visit to talk and eat. It's not clear how to get them to understand the complexity and they will not get to speak to every professor they requested a meeting with.
- Faculty like to eat, talk to some students they want to and don't get to talk to some students they wanted to, and ask to meet with students who don't visit.
- Current students get little done on their theses, get to eat and talk a lot to students who may have really wanted to talk to their advisor or may have no common research interest at all.
- Faculty who are not in the Computer Science Department proper generally have little vested interest in meeting with a student during your Open House and may or may not offer time to do so, thus increasing the complexity factor by the number faculty requested by admits from departments outside yours.
- Some faculty who are not in the Computer Science Department proper may be more accommodating with their time than faculty in your own department, thus increasing the complexity factor by the number faculty requested by admits from departments outside yours.
- Current students are helpful and a tremendous benefit to have on hand

[1] We didn't have time to explore these in detail and apply them to our optimization processes due to the following factors: (1) faculty making their own appointments caused an additional time sink and domino affect on scheduling that had to be handled before (2) applying the requests of admitted students after they arrived and wanted additional faculty added to their schedules while (3) adding faculty who were suggested by other faculty to admitted students schedules while (4) changing counts for events that students would now not be attending due to additional appointments added to their schedule while (5) simultaneously sending email to multiple admitted students and faculty to confirm changed or additional appointments while (6) updating, printing and distributing physical schedules and (7) contacting the current students to either move appointments with an admit or (8) emailing current students to cancel appointments entirely then (8) repeat the process when faculty decided at the end of day one that they did actually want to speak to another student on day two of the event even though they originally said they had no available time on day two while (9) contacting admits to cancel appointments with faculty unable to return to campus due to travel issues while (10) redoing schedules for any admitted students who had travel issues.

for the events and appointments and scheduling couldn't truly be accomplished without their willingness to fill in for busier, distracted, or missing faculty.
- Some faculty need to learn how to say "No".
- Some faculty need to learn how to say "Yes".

While the research is not at all conclusive about the best optimization methods to apply to the herding of cats[2], it is apparent that more research is needed so admissions coordinators can optimize the scheduling (if there is time and if there is funding leftover from the food and admitted student travel budgets).

Without the possibility of applying optimization for this process, each incremental increase in the number of admits attempting visit, crossed with the number of faculty and departments engaged in the open house process will steadily head toward rift creation. With the addition of each request to meet with a faculty member outside the hosting department coupled with the admits expectations crossed with the amount of time actually available to accomplish concurrent tasks the potential escalates.

Questions we have not yet answered satisfactorily and might be suitable for future research are:

- Should we give the admitted students t-shirts? Everybody wants another t-shirt! Our research leads us to believe we should – perhaps to keep up with the "Joneses"[3](you know, those other schools that think they are top-tier and just may have handed out t-shirts when we did not) or to add something else neat to the gift bags.
- Should we build a water park with a huge beach as a recruiting tool, closer to campus than Sandcastle, of course. (everyone wants to be closer to more water than just 3 rivers and there should be lots of sand, right - Sandcastle is just too far away - seriously!)?
- Should we build a solar dome to enable suntanning in the winter months to compete with the West Coast schools?
- Should we somehow encourage the development (or build one ourselves) of more ski areas within driving distance to take advantage of the fact we seem to be drifting north as far as weather patterns are concerned? (Look out East Coast schools north of us - we are about to seriously compete with your horrible weather!)
- How much email was actually sent and received by the admissions coordinator during the time period of notifying students they were admitted, their Open House travel planning, attendance and reimbursements and the decision deadline?
- Determining if coffee consumption is a valid method to sustain the process and stave off the pending multidimensional space-time rift.

IV. Discussion

Other than a few long and many hurried meetings with the one person who created this process, there has not been any discussion about the overall dilemma regarding our potential to cause a multi-dimensional space-time rift. Input would be welcome regarding methodology. Not that I would apply any of them to future research, but it might be a fun discussion nonethe-less.

It is hoped that discussion will not be terribly serious, as this research was done with a lighthearted spirit. Data is still coming in and the processes the are ongoing. I look forward to the next round of applications, admissions meetings and Open House as sources of continued research. ;-)

[2]This is a classic dilemma that surfaces repeatedly and is almost lovingly referred to by administrators across academia[4] see also: `http://chronicle.com/blognetwork/researchcentered/2011/09/23/advanced-faculty-wrangling-techniques/`

[3]Wikipedia definition `http://en.wikipedia.org/wiki/Keeping_up_with_the_Joneses`

References

[1] P. Bar-Joseph, *Space-time discontinuous finite element approximations for multi-dimensional nonlinear hyperbolic systems*.

Computational Mechanics, Volume 5, Issue 2-3, 145–160, Springer-Verlag, 1989

[2] Vincent Conitzer, Tuomas Sandholm, *Compute the Optimal Strategy to Commit to*.

EC '06 Proceedings of the 7th ACM conference on Electronic commerce, 82–90, ACM 2006

[3] Carlos Eduardo Paiva, Joao Paulo da Silveira, Nogueira Lima, and Bianca Sakamoto Ribeiro Paiva *Articles with short titles describing the results are cited more often*.

Clinics (Sao Paulo) v.67(5); May 2012

[4] Marietta Del Favero, Nathaniel J. Bray Ph.D., M.Ed., *Herding Cats and Big Dogs: Tensions in the Faculty-Administrator Relationship*

Higher Education: Handbook of Theory and Research, 477-54, Springer Netherlands, 2010

Track 888

New Theoretical Perspectives

1. **Heterotopy Type Theory:**
   A Defense of the Traditional Foundations of Mathematics
   Thomas Cramer and Sam Yeager
   Keywords: heterotopy type theory, identity type, freedom of expression axiom

2. **The Dumping Lemma – Assessing Regularity**
   Naomi Saphra
   Keywords: computational scatology, formal languages, poop jokes

3. **Statistical Watch Optimization**
   Bryce Summers
   Keywords: Watch, clock, time, probability, statistics

4. **A Simple Category-Theoretic Understanding of Category-Theoretic Diagrams**
   Stefan Muller
   Keywords: category, diagram, functor, diagram diagram

Heterotopy Type Theory: A defense of the traditional foundations of mathematics

Thomas Cramer (R-TX)
United States Senate

Sam Yeager
Arizona State Senate

Abstract

A major focus of interest in type theory has been the treatment of *identity types*, a subject that recently underwent a major revision under homotopy type theory [4]. Researchers in the field have historically ignored the beliefs of a large segment of the population that wishes to preserve the traditional meaning of identity types. Many authors mistakenly refer to these types, even informally, as "proofs of equality." This is not about equality. It is about our freedom to practice type theory as we see fit, and we will defend this freedom against any onslaught of political, or mathematical, correctness. This paper takes a firm stand against the liberal attack on identity, instead offering a pure account of identity types, called *heterotopy type theory*, which reaffirms their original intent. We hope this document will be enshrined as a part of the Constitution of the Association for Computing Machinery [1], to protect the august institution of identity against future attacks.

1. Introduction

Much prior work in type theory has dealt with the ways in which type theories handle identity types $\mathsf{Id}_A(m,n)$, where A is a type and m and n are terms of that type. With the recent introduction of homotopy type theory [4], the discussion of these identity types has gained renewed interest, leading to a total redefinition of the meaning of identity. Under this disturbing philosophy, terms of identity type are thought of as paths in the space representing the type A. Elements of the iterated identity type $\mathsf{Id}_{\mathsf{Id}_A(m,n)}(p,q)$ where p and q are paths between m and n may be thought of as *homotopies* between these two paths. But it is not for us to redefine identity. Identity is a concept that has been steeped in tradition since the beginning of our field, and is granted a unique status in the ancient books of mathematics. It may not be lightly altered to fit every passing trend, however fervent the supporters of this trend may be.

2. Traditional identity types

The primary problem with new definitions of identity is the so-called *identity introduction* rule.

$$\textsf{Id-I-UNNATURAL}\quad \frac{\Gamma \vdash m : A}{\Gamma \vdash \mathsf{refl}_A(m) : \mathsf{Id}_A(m,m)}$$

This rule embodies the natural and logical conclusion of the slippery slope on which type theory finds itself: if two things of similar nature are allowed to form an identity, then why not allow a term to enter into an identity with itself? The identity "proof" refl_A represents all that is dangerous about modern type theory, and we should not allow ourselves to tend in that direction. We, of course, do not allow such a rule within our type theory. Instead, we propose an alternate identity introduction form, which we call *original identification* and notate $\mathsf{ido}^F_A(m,n)$. To make sure that this form cannot be twisted to allow instances of refl, it requires a witness, $F : A \to \mathsf{bool}$, which asserts that m and n are eligible to be identified by mapping one to tt and one to ff.[1]

$$\textsf{Id-I}\quad \frac{\Gamma \vdash m : A \qquad \Gamma \vdash n : A \qquad F(m) = \mathsf{tt} \qquad F(n) = \mathsf{ff}}{\Gamma \vdash \mathsf{ido}^F_A(m,n) : \mathsf{Id}_A(m,n)}$$

This is what one would expect as the natural definition of identity. After all, it was Adam and Eve, not Adam and λx. Adam x. Unfortunately, removing reflexivity in favor of our traditional introduction form is not sufficient to restore traditional identity. The depravity of modern type theory continues to such an extent that the following term is actually *definable* within the type theory:

$$\mathsf{trans} : \mathsf{Id}_A(l,m) \to \mathsf{Id}_A(m,n) \to \mathsf{Id}_A(l,n)$$

While some would have us believe that trans is a perfectly natural and wholesome property, we will see where this leads: having entered into a solemn identity with l, m is immediately allowed to enter into an identity with n as well. Were this not enough, trans has the audacity to suggest that l and n are now identified! Clearly we cannot allow such an

[1] Many types actually require two witnesses, but we present only one here for simplicity.

abomination to enter into our system. The root of this danger is identity elimination, the so-called J rule.

$$\textsf{Id-E-Unnatural}\quad \frac{\begin{array}{c}\Gamma \vdash p : \mathsf{Id}_A(M,N) \\ \Gamma, x:A, y:A, z:\mathsf{Id}_A(x,y) \vdash C\ \mathsf{type} \\ \Gamma, x:A \vdash Q : [x, x, \mathsf{refl}_A(x)/x, y, z]C\end{array}}{\Gamma \vdash \mathsf{J}[x.y.z.C](p, x.Q) : [M, N, p/x, y, z]C}$$

This rule demonstrates the unbounded audacity of proponents of this unnatural and dangerous type theory. It is not enough for them that a concept like refl be introduced into the theory, but they must impose it on all type theorists by asserting an elimination form that treats all self-respecting identities as if they were refl! As we did with identity introduction, we must present a proper identity elimination form, which we call H after incisive social commentator Sean Hannity. This rule states that once an identity p between x and y has been sanctified by any witness, we may use it in any context C requiring identified elements without knowing what witness was used. In particular, we may do this as long as we can form a term Q that is typed with C assuming that x and y were identified by a witness that assigns tt to x and ff to all other elements of A.

$$\textsf{Id-E}\quad \frac{\begin{array}{c}\Gamma \vdash p : \mathsf{Id}_A(M,N) \\ \Gamma, x:A, y:A, z:\mathsf{Id}_A(x,y) \vdash C\ \mathsf{type} \\ \Gamma, x:A, y:A \vdash Q : [x, y, \mathsf{ido}_A^{\mathbf{1}_{\{x\}}}(x,y)/x, y, z]C\end{array}}{\Gamma \vdash \mathsf{H}[x.y.z.C](p, x.y.Q) : [M, N, p/x, y, z]C}$$

where $\mathbf{1}_{\{x\}}$ is an *indicator function*

$$\mathbf{1}_{\{x\}}(z) = \begin{cases} \mathsf{tt} & : z = x \\ \mathsf{ff} & : z \neq x \end{cases}$$

Homotopy type theorists might, at this point, display their inner doubt with the system they have created by attempting to justify their inference rules with a "local soundness proof." We will not do this here, as our faith in the goodness of traditional identity types is complete and no other proof is required.

Lest we leave the reader without hope, we should note that one tenet of our philosophy remains uncorrupted by the forces against which we struggle. The term $\mathsf{sym} : \mathsf{Id}_A(x,y) \to \mathsf{Id}_A(y,x)$, which recognizes the mutual commitment to the bond of identification, continues to have a place in modern type theory. Of course, sym is definable in our theory as well.

$$\mathsf{sym}\ x\ y\ p := \mathsf{H}[x, y, \_.\mathsf{Id}_A(y,x)](p, x.y.\mathsf{ido}_A^{\mathbf{1}_{\{y\}}}(y,x))$$

3. Functoriality

Our efforts to correct the ills of type theory are thwarted once again by the functorial action ap. Suppose we have a map $F : A \to B$, and two elements x, y such that $\Gamma \vdash x : A$ and $\Gamma \vdash y : A$. Define ap_F to be the action on paths of F. Activist type theorists would like us to blindly assign to ap_F the type $\mathsf{Id}_A(x,y) \to \mathsf{Id}_B(F\ x, F\ y)$. But suppose that some corrupting influence causes it to be the case that $F\ x = F\ y$. Even if we have an honest-to-goodness identity $p : \mathsf{Id}_A(x,y)$, such as one formed by ido and a proper witness, $\mathsf{ap}_F\ p$ will be typed by the immoral and unnatural type $\mathsf{Id}_a(F\ x, F\ x)$. But all is not lost. If F is injective, we can be sure that the type B will not diminish the identification of x and y by allowing their F-mapped counterparts to live in refl. Therefore, when identifications are mapped across types by functorial actions, it is important that we only continue to accept the identification if the map in question is a good, injective map. Recent work by Kennedy, et al. [5] suggests that the action on paths of non-injective maps might be recognized at a higher universe encompassing both A and B, but we do not accept the validity of this work.

4. Higher inductive types

Identities hold such an exalted role within our society that it seems natural to allow them to be defined directly as part of inductive types. For example, the following is a definition of the type bool, which not only defines the two elements, tt and ff, but allows them to enter into a natural, traditional, identity that reaffirms their commitment to be partners in their membership of the type bool:

$$\begin{array}{rcl} \mathsf{tt} & : & \mathsf{bool} \\ \mathsf{ff} & : & \mathsf{bool} \\ p & : & \mathsf{Id}_{\mathsf{bool}}(\mathsf{tt}, \mathsf{ff}) \end{array}$$

Non-believers might not accept such a definition that asserts an identity without what they would call "proof." Yet it is a fundamental facet of heterotopy type theory that we must take certain definitions on faith; this is why they are called *Higher* inductive definitions.

5. Freedom of Expression Axiom

On the other hand, it is sometimes the case that we will be presented with an alleged identification that must not be considered valid, as to do so would conflict with our most deeply-held beliefs. To make sure that others cannot infringe upon our rights by forcing us to accept improper identifications, we introduce the Freedom of Expression Axiom. This axiom provides a term $\mathsf{fea}_A(x,y)$ for x, y of type A, which contradicts such a destructive identification.

$$\textsf{FEA}\quad \frac{\Gamma \vdash x : A \qquad \Gamma \vdash y : A}{\Gamma \vdash \mathsf{fea}_A(x,y) : \neg\mathsf{Id}_A(x,y)}$$

6. Related Work

The most closely related work to this paper in its free, natural approach to type theory is Angiuli's work on unintentional

type theory [2]. However, we resent that Angiuli, who has previously embraced socialist policies [3], would refer to this type theory as "unintentional" when it is clearly part of a larger, Intelligently Designed type theory.

References

[1] Constitution of the ACM. `http://www.acm.org/about/constitution`, 2014.

[2] C. Angiuli. The $(\infty, 1)$-accidentopos model of unintentional type theory. In *Proceedings of the 7th annual intercalary robot dance party in celebtration of workshop on symposium about Harry Q. Bovik's* 2^6*th birthday (SIGBOVIK '13)*. ACH, Apr. 2013.

[3] K. Angiuli and F. Engels. Redistributive version control systems. In *Proceedings of the 7th annual intercalary robot dance party in celebtration of workshop on symposium about Harry Q. Bovik's* 2^6*th birthday (SIGBOVIK '13)*. ACH, Apr. 2013.

[4] I. for Advanced Study. *Homotopy Type Theory: Univalent Foundations of Mathematics*. The Univalent Foundations Program, 2013. `http://homotopytypetheory.org/book/`.

[5] A. Kennedy et al. United States v. Windsor. *570 U.S. 12 (Docket No. 12-307)*, 2013.

The Dumping Lemma
Assessing Regularity

Naomi Saphra

Abstract

In the field of computational scatology, there has long stood a question of how one might approach the question of a language's regularity. It is widely acknowledged as desirable that a language should be regular, as this eases the generation of outputs as well as the digestion of inputs. However, it is often not obvious whether a given language is, in fact, regular or tractable. Here we describe the use of the Dumping Lemma to prove the regularity of a language.

1 Introduction

Because computational scatology is primarily concerned with outputs, the most popular models in the field tend to be generative. Since the field's humble beginnings, our concern with the taxonomy of output languages and sets according to the automata that might have generated them has been paramount. Once "Father of Computational Scatology" Anal Püring first proposed the taxonomy that would bear his name of Püring-computable and -complete languages, we developed vocabulary to describe further demarcations of language categories. Among our family of automata are the relatively simple models that can generate or recognize the regular languages, Fecal State Automata (FSA) (as well as their relatives, the Fecal State Trans**poo**cers, beyond the scope of this paper).

Unfortunately, the coprocomputability community has struggled to consistently classify a given output set as regular or not. Thus we present a new rule that can be employed in proving a regularity claim.

2 The High-Fiber Hypothesis

In our work with FSAs, we have encountered a pattern of high-fiber inputs to a recognizing automaton. This leads us to the conclusion that regular languages must have a high-fiber segment, that is:
A regular input longer than length k must have a segment of fiber that can be expanded arbitrarily and the input is still accepted by an FSA.

This is the claim that we will call the **Dumping Lemma**.

Proof In Figure 1 are images of regular inputs sets that somehow we consider tantamount to a constructive proof. The fecoinformatic implications of the evidence are obvious and left as an exercise to the reader.

3 Future Work

We believe there to be a possible similar lemma for the higher-order category of Context Pee Languages (CPLs). Though their context is pee, they share many qualities with RLs, as

Figure 1: Regularizers

they can be generated by Fecal Tree Automata. Because a feces tree has never been observed in the wild or implemented in practice, we cannot confirm these suspicions, but do present it as a possible path to a higher citation count. It is worth noting that trees are widely considered to metabolize by excreting oxygen and water, posing a further challenge for any attempt to prove a Context Pee variant of the Dumping Lemma.

Statistical Watch Optimization

Bryce Summers
whowatchesthe@mailman.cmu.edu
SIG BOVIK Conference, April 1, 2014
March 7, 2014

Disclaimer

This research was made possible in part by generous funding from 1 - 800 - JUNK and the Combinatorics, Linear Optimization, and Cool Kinematics research foundations. All information can be corroborated by Father Time and therefore no source need be cited.

Abstract

In this paper we will be demonstrating applications of statistics and probability theory to the reduction of mechanical energy used by the clocks and watches that inhabit our world. We have found that we can yield as much as 100 percent reductions in energy usage by breaking all of the watches and that their functionality will not be affected.

Broken Watches are Optimal.

We must initially prove some complicated lemmas, and only then will we be able to get to the intuitive proof.

Lemma 1 : Watches are either working or broken

There has been much debate over the years as an exact classification of watches. Some watches perform full double pi radian oscillatory movement, whereas others display dynamically induced interrupt functionality. The watches that display distortion within their homogeneity fields may have different magnitudes to their temporal displacement vectors. While fully broken watches display fully disrupted behavior, mongrels of the partial broken category may still provide oscillatory movement that is not at ease with classical harmonic rhythms expected from the modern day busy intellectual. For our purposes we will simplify our analysis to those watches that display full and nonexistent disruptive behavior, where we will call the non-disrupted watches **working** and the fully disrupted watches **broken**

Lemma 2 : Working watches maintain a constant have a constant error ϵ in their temporal lookup provider functionality.

We need to show that working watches have a non-zero error from real time and that they retain this error until the end of time.

First of all we know that watches are set through mechanical machines or human hands and that they will not be set to the exact correct positions due to the vibrations of the temporal setting apparatus.

Second of all, since we are considering theoretical lambertian watch faces, the light scattering on the watches' temporal lookup oscillators can be safely ignored and we can assume that working watches will maintain a constant ϵ error from the real time specified by the temporal laws of our forefathers since the dawn of time.

We can prove this second part via induction:

Base Case: The claim holds now.
Induction Hypothesis: The claim holds before now'.
Induction Step: Since the nature of time is subjective, and nothing really happens in an infinitely small amount of time, the claim must hold now'.

Lemma 3 : Broken Watches are correct two times a day.

Due to the closure of the real numbers mapped radians under the theoretically correct movement operation of a perfect watch oscillatory component, every point on the watch must be hit twice through the span on 1 day. Since a broken watch must exhibit 1 constant state throughout the day, it must exactly match the correct state precisely two times in every day.

Lemma 4: Errors in time are signed values.

Humans naturally discuss errors in time using signed values. For instance, 4:01 pm would be said to be 1 minute **after** 4 instead of 59 minutes **before** 5. Such a distinction of the two signed categories of "after" and "before" is a natural convention in the understanding of numerical time.

Lemma 5 : AM and PM are identical on mechanical watches

Watches providing movement of lack thereof through mechanical means are assumed to provide the same user experience in the first half of the day as the second.

Lemma 6 : Broken Watches need less energy to operate than Working Watches.

Due to the abscense of movement in Broken Watches, they can be implemented without the use of any mechanical motor, whereas Working watches need to have energy sources installed to facilitate the movement of their temporal reading components. Therefore, Broken Watches consume less energy than working watches.

Broken Watch Optimality:

Many people are under the impression that Working Watches are more useful for telling time than Broken Watches. We will attempt to examine potential why people might come to this fallacious reasoning and attempt to help these lost souls see the virtues of Broken Watch utilization.

Fallacy 1: Correctness.
Many rational people believe that Working watches are more correct than broken watches. This is simply not the case. Lemma 2 shows that Working watches have a finite error at every time during the day, so they are correct 0 times in every day. Lemma 3 shows that Broken Watches are correct two times a day.

Fallacy 1: Expected Error.
A perceptive person would probably state that the relative minuteness of the size of errors in a temporal information provider is much more important that absolute correctness. Such as person would cite lemma 2 to show that Working watches have an expected error of ϵ. They would then in their hubris neglect to examine the case for broken watches, believing they have proved the optimality of Working watches. If such a narrow-minded individual were to actually compute the expected value of the average error for a Broken watch they would learn the expected error of their ways.

By Lemma 4, A broken watch goes linearly from being 0 hours off to 12 hours off to - 12 hours off to

0 hours off during the course of a day. Therefore we can compute the expected error as follows:

$$\mathbb{E}[Broken\_Error] = \frac{\int_0^{12} t\mathrm{dt} + \int_0^{12} -t\mathrm{dt}}{24} = 0$$

As witnessed by the impressive calculus, the Broken watch is expected to on average be exactly correct. Thus, the Broken watch is on average less erroneous than a working watch.

Fallacy 1: Military Time Expected Error.
An astute observer may object to our logical use of signed time and demand that we hand error more precisely with less room for creative interpretation.

In this case, we must yield to their demands and analyze the expected error under military time, the most rigorous of time demains.

In this domain the error of the Working watch will still be ϵ by lemma 2.

The Broken watch's error would vary uniformly from 0 hours to 24 hours. Thus we compute the expected error as follows:

$$\mathbb{E}[Broken\_Error] = \frac{\int_0^{24} t\mathrm{dt}}{24} = 12$$

So by the groovy calculus, the expected error for the Broken Watch is 12 hours. By lemma 5, we can conclude that a rational individual should expect the watch to be exactly correct.

Thus, under the strict standards of military time Broken Watches still exhibit optimality over their working counterparts.

Energy Savings:

By lemma 6, transforming all of the working watches in this world into broken watches, will lead to savings of $4 \cdot \pi$ radians of mechanical energy per watch per day and thereby help reduce the worlds' energy consumption, while providing users with devices that have superior correctness and error reduction.

Conclusion:

As our mathematics demonstrates, the current reliance of stressed out individuals on concepts such as time, productivity, and watches may be profoundly misguided and that many of the fallacious institutions that we take for granted should be reexamined and optimized using proper probabilistic methods.

SIGBOVIK 2014 Paper Review

Paper 8: Statistical Watch Optimization

Ben Blum, Light Cone Sedentarian
Rating: 3 (weak accept)
Confidence: 4/4

The author presents an innovative new method for maintaining watch accuracy that requires unprecedentedly low amounts of maintenance energy. However, I must note two major flaws with this work:

1. The author fails to survey related work, most importantly, the seminal publication in the field "Mustard Watches", published well in advance of this very conference's founding.
2. The author's treatment does not account for the effects of relativity on broken watches. In modern times, time-travellers will have been breaking causality left and right, and it is of paramount importance for average citizens to be able to communicate with them if one is wearing a broken watch and the other is not.

Nevertheless, this is an important theoretical advancement in the field of Watch Theory, and opens several promising directions for future and past work. Weak accept.

A Simple Category-Theoretic Understanding of Category-Theoretic Diagrams

Stefan Muller
Carnegie Mellon University
smuller@cs.cmu.edu

Abstract

Textbooks and other introductions to topics in category theory often use the *commutative diagram* as a convenient tool to explain new ideas. Unfortunately, for the uninitiated, these diagrams can cause more confusion than enlightenment. In this paper, we seek to give a gentle introduction to the use of commutative diagrams. To simply and clearly explain this important concept, we frame the discussion in terms of simple category theory itself and use a convenient tool for understanding new ideas in category theory: commutative diagrams.

1. Introduction

Many discussions of category theory graphically represent the objects and morphisms of a category using *diagrams*. In a diagram, objects of a category are represented using one or more glyphs, typically a single uppercase letter of the Roman alphabet. A morphism between objects is represented using a straight or, in rare instances where it is required for clear planar presentation of the diagram, curved, line also labelled with one or more glyphs, typically a single lowercase letter of the Roman alphabet. Relationships between two paths between objects are indicated by the commutativity of the diagram. While to the trained category theorist, a commutative diagram can convey a great deal of information about an unfamiliar category under discussion, students and those in other fields often have trouble understanding the relationships implied by a commuting diagram. It is often possible to explain an individual diagram in prose, but this defeats the purpose of the simple graphical representation. This paper instead aims to explain diagrams in general by presenting these diagrams as functors. It is known that a diagram of a category C may be viewed as a functor $F : D \to C$ where D is a small category (e.g. [1]), but this paper chooses an alternate presentation that we believe is unique in its clarity and concision.

2. Diagrams as a Functor

Let C be a small finite category. We define a functor D from C to diagrams of C. For an object $A \in C$, $D(C)$ is the diagram

$$A \xrightarrow{f} B$$

Figure 1. The diagram $D(f)$ of the morphism f.

$$A \xrightarrow{f} B \xrightarrow{g} E$$

Figure 2. Composition of the morphism diagrams $D(f)$ and $D(g)$.

$$A$$

For objects $A, B \in C$, we can take the diagram of a morphism $f : A \to B$. $D(f)$ is shown in Figure 1.

If $A, B, E \in C$ and there are two morphisms $f : A \to B$ and $g : B \to E$, the diagrams $D(f)$ and $D(g)$ may be composed by merging them together at their common nodes, in this case, B. $D(g) \circ D(f)$ is shown in figure 2.

We now show that D is a functor. Let $A \in C$, and id_A be the identity morphism on A. $D(\mathsf{id}_A)$ is shown below.

$$A$$

It is obvious from the definition of morphism diagram composition that this is the identity morphism on $D(A)$. We must next check that, for $A, B, E \in C$, $f : A \to B$ and $g : B \to E$, we have $D(g \circ f) = D(g) \circ D(f)$. This fact is clear from the commutativity of the diagram in Figure 3.

3. Morphisms on Diagrams

We now observe that, for a category C, $D(C)$ forms a category of diagrams on C. Let C be a category. We define an object in $D(C)$, $\mathcal{A}$:

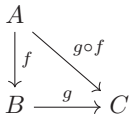


Figure 3. D preserves composition.

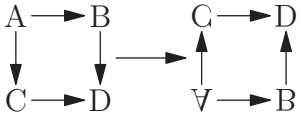


Figure 4. The morphism $\downarrow$.

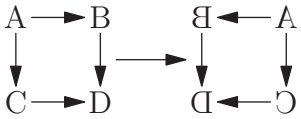


Figure 5. The morphism $\leftarrow$.

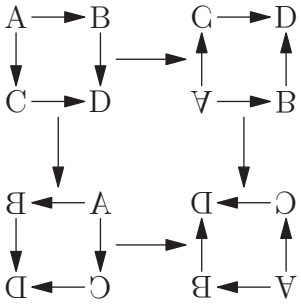
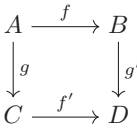


Figure 6. Commutativity of $\downarrow$ and $\leftarrow$.



One morphism $\downarrow$ is shown in Figure 4. Another simple morphism, $\leftarrow$, is shown in Figure 5. $\leftarrow \circ \downarrow = \downarrow \circ \leftarrow$, as shown by the commutativity of Figure 6.

4. The Functor $\circlearrowleft$

We now describe a functor $\circlearrowleft$ on the category $D(C)$ for any small category C. $\circlearrowleft(\mathcal{A})$ is defined to be the diagram in Figure 6. We refer to $\circlearrowleft(D(C))$ as the category of *diagram diagrams* on C. Since diagram diagrams are also diagrams,

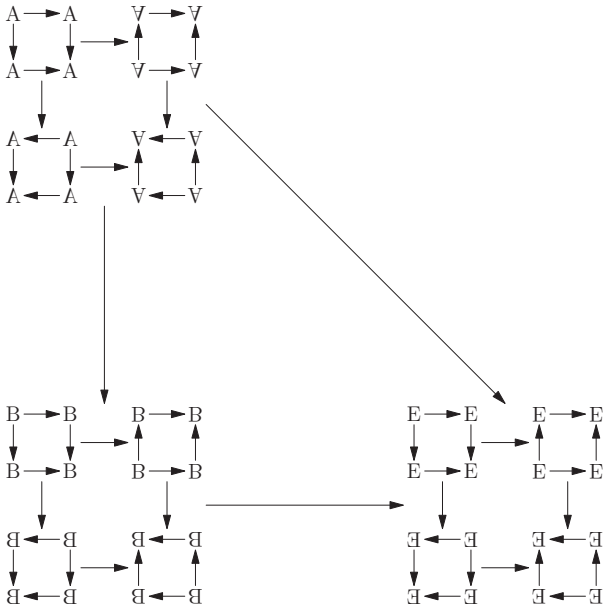


Figure 7. $\circlearrowleft$ respects composition.

the same morphisms apply. For instance, $\circlearrowleft(\leftarrow) = \leftarrow$ and $\circlearrowleft(\downarrow) = \downarrow$. We also have that, for any $\mathcal{A} \in D(C)$, $\circlearrowleft(\mathsf{id}_{\mathcal{A}}) = \mathsf{id}_{\circlearrowleft(\mathcal{A})}$. It now remains to check that if we have $\mathcal{A}, \mathcal{B}, \mathcal{E} \in D(C)$, $f : \mathcal{A} \to \mathcal{B}$ and $g : \mathcal{B} \to \mathcal{E}$, then $\circlearrowleft(g \circ f) = \circlearrowleft(g) \circ \circlearrowleft(f)$. This is shown by the commutativity of the diagram in Figure 7. We may now demonstrate for $\circlearrowleft$ all of the standard properties of functors, one of which is shown in Figure 8.

5. Future Work

We hypothesize, but have not yet shown, that the functor $\circlearrowleft$ can be nested arbitrarily deeply, resulting in a family of functors $\circlearrowleft_i$. A conception of $\circlearrowleft_4$ is shown in Figure 9. Understanding of the limit of this process, notated $\circlearrowleft_\infty$, will provide a fuller understanding and newfound clarity to the device of commutative diagrams.

6. Conclusion

This paper has shown by example that it is simple to gain a thorough understanding of an unfamiliar topic in category theory through its presentation in terms of commutative diagrams. We hope that this work will be useful in the future to beginning students of category theory and related fields of study. While commutative diagrams are just one path to a full understanding of any concept, they are equivalent to any other such path. We hypothesize that this fact can be demonstrated using a commutative diagram.

References

[1] J. P. May. *A Concise Course in Algebraic Topology*. The University of Chicago Press, Chicago, 1999.

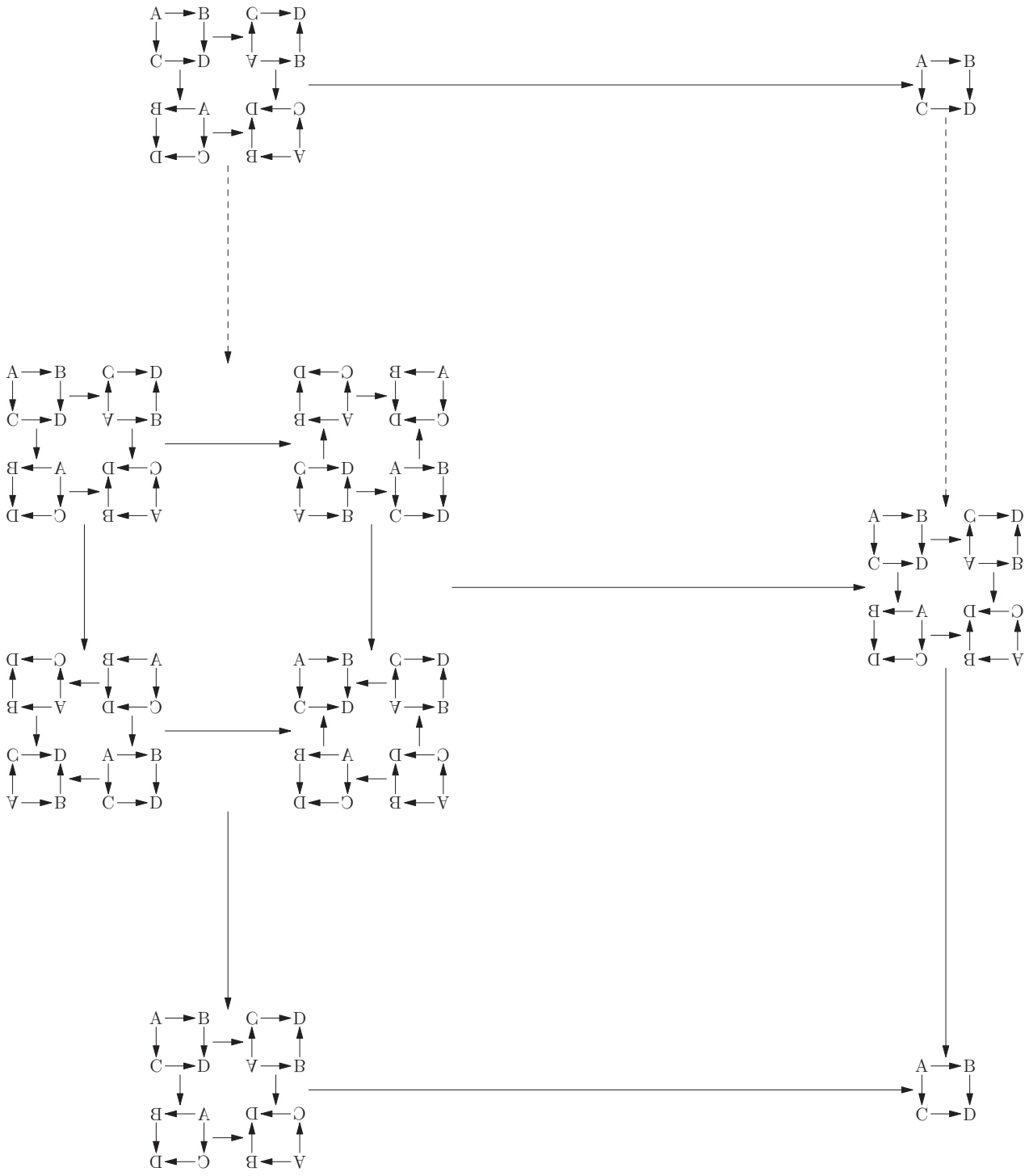


Figure 8. A trivial property of the functor $\circlearrowleft$.

Figure 9. A conjecture as to the structure of $\circlearrowleft_4$.

SIGBOVIK 2014 Paper Review

Paper 9: A Simple Category-Theoretic ...

Reviewed by: Ed Morehouse (Carnegie Mellon University)

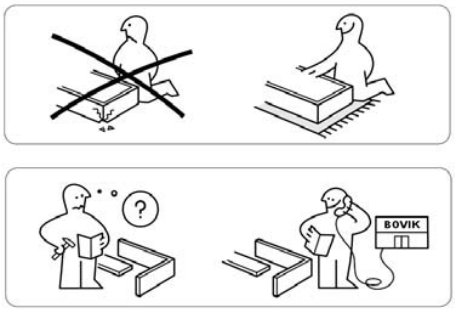
"A Simple Category-Theoretic Understanding of Category-Theoretic Diagrams" by "Stefan Muller" is a work of incomprehensible abstract nonsense, and as such, constitutes a valuable contribution to the field of category theory.

The article seems to be about understanding category theory through the graphical language of commutative diagrams. I say, "seems to be", because after gleaning this much from the abstract, I decided to test (what I assume to be) the article's central thesis by attempting to understand the article itself by just looking at the pictures.

I infer that the article begins by describing the basic properties of morphisms in a category under composition and identity – although it could be talking about something else entirely, it's kind of hard to say. The next bit seems like it might be about constructing a category whose objects are themselves commutative diagrams and whose morphisms are affine transformations of such[1].

The paper ends by constructing the "Morehouse-Sierpinski ω-category of commuting squares" (let's call it). Although "Muller" has (for all I know) now solved a long-standing open problem in higher dimensional category theory by giving a finitary, constructive presentation of this important category, there is still currently no type-theoretic interpretation (that I could think of in 5 minutes), nor are current LaTeX diagram-description packages adequate for drawing ω-dimensional commutative diagrams. I'll just assume that "Muller" leaves these issues for future work.

Rating: I give this article the terminal rating in the co-op of the bicategory of journal reviews.
Confidence: What am I supposed to put here?
Why are the instructions not presented in picture-form, like the ones for assembling my bookshelf:



[1] note to self: ask "Muller" what the paper is really about, then reject it and steal his idea.

Track 8,888

Stay Safe Out There

1. **Cryptographically-Sound Jokes**
   Jim McCann
   Keywords: jokes, sha, md5

2. **DollarCoin: A Cryptocurrency with Proof-of-Dollar**
   Patrick J. Xia
   Keywords: cryptocurrency, bitcoin, dollars, bills, gamechanger, disrupt, awesome, supercool, thebest, hashtag

3. **Towards a Completely Autonomous Vehicle**
   Nicholas Fudala and Chester Francis
   Keywords: pain, suffering, frustration

Cryptographically-sound Jokes

Jim McCann*
TCHOW llc

What do you call a predatory fish endorsed by the NIST and NSA?

– A 6265a4a07968a7c1df16a61004fb7191177cafd4[†]!

Did you hear that the NSA is going into manufacturing collision-resistant form-molding undergarments?

– Yeah, and they are changing their acronym to "NDA" for New d4193f19ed359e122dbf51cc81e94145ca017bc8[‡] Associates.

How do you tell a member of the cryptographic elite from the plebes?

– Ask them to pronounce b43c9a57aaf61f43e5021d305146d21473bded99[§]!

Why did the drug user choose 9b4e473e4de7949c4ab8e441a7faa5e2b295b469[¶] over 4a82715423d654d61838e81060a4cdf1[‖]?

– Because they always take the bag with the strongest hash.

What do you call an strongly mixed certifcate of ownership?

– c387c982a132d05cbd5f88840aef2c8157740049[**], of course!

*e-mail: ix@tchow.com
†`SHA('rk')`
‡`SHA('pewear')`
§`SHA('bboleth')`
¶`SHA('bag')`
‖`MD5('bag')`
**`SHA('re')`

SIGBOVIK 2014 Paper Review

Paper 2: Cryptographically-sound Jokes

Emily Forney, The Humanities in General
Rating: 88 (strong accept)
Confidence: 1/4

I really enjoy that the answers are little and at the bottom. If they were larger and more noticeable it might have spoiled the jokes which I totally figured out on my own. They also had really cute little symbols to denote which joke they match up with. Not that I needed that.

I strongly accepted this as a collection of jokes that are both about cryptography and also written in cryptography, you know, the language. That is completely in line with the title.

My confidence reflects my understanding of the topic: a 1, as in the best level of understanding possible. Which is what I have.

We share jokes, we are friends now.

DOLLARCOIN: A CRYPTOCURRENCY WITH PROOF-OF-DOLLAR

PATRICK J. XIA

Abstract. We introduce DollarCoin, a cryptocurrency that uses a novel scheme, proof-of-dollar, to achieve a consistent and secure blockchain. Security of the currency does not require energy consumption or useless "make work" hash computations, as with extant proof-of-work schemes, nor is the scheme vulnerable to verification monopolies by early adopters, as is the case with proof-of-stake schemes. Security of the blockchain is instead regulated by the relative scarcity of the United States Dollar. DollarCoin is the first cryptocurrency that directly satisfies all three properties of currency: it is a medium of exchange, a unit of account, and a store of value.

Keywords. cryptocurrency, bitcoin, dollars, bills, gamechanger, disrupt, awesome, supercool, thebest, hashtag

1. Introduction

Modern cryptocurrencies use a disincentive-based system to make changing the network consensus economically difficult. In the case of Bitcoin, the specific scheme used is the computation of a partial hash inversion. Since there are, to date, no breaks in the SHA-256 cryptographic hash system, the best way to compute such a partial hash inversion is by brute-forcing until one gets lucky and finds a difficult "proof of work"; specifically, a extremely small (and therefore unlikely) output hash value. This secures the blockchain as long as nobody controls a large share of the computation power of the network, but at the cost of an arms race as people attempt to compute hashes as quickly as possible to increase the chance of finding a "block." Such Bitcoin "mining," as it is called, requires a great deal of energy expenditure to keep the network secure.

We propose skipping the middleman and providing direct proof-of-dollar with a blockchain that consists of videos of burning US $1 notes. Each video will be unique as the hash of every block header is required to be written on the dollar bill to be burnt; the block header includes the previous block's hash, which means that each block is forced to build on every previous block, forming a valid blockchain.

2. Implementation

DollarCoin is implemented identically to Bitcoin with the exception of block format, as no nonce or difficulty adjustment is required. The transaction format remains the exact same so that people may use extant code to manually generate transactions for the DollarCoin blockchain. The same system of double-SHA-256 hashes ensures cryptographic integrity of the blockchain. A block contains these fields:

Thanks to Harry Quetzalcoatl Bovik.

Field	Description	Size (octets)
Magic Number	Value always 0xD9B4BF00 (one more than Bitcoin)	4
Block size	Number of bytes until end of block	4
Block header	Consists of block header, described below	72
Transaction counter	Variable length positive integer (Bitcoin VarInt)	1 - 9
Video link length	Number of bytes required for the video link	2
Video link	A (possibly transient) link to a video	?
Transactions	The list of transactions	?

and the block header is specified as so:

Field	Description	Size (octets)
Version	Block version information	4
Previous block	Previous block's hash value	32
Merkle root	The hash of the Merkle tree of the transactions	32
Timestamp	Timestamp recording when this block was created	4

All hashes are the double SHA-256 scheme as specified in Bitcoin. Block header hashes are independent from the video link, allowing for video links to be changed in peer to peer communications so that they are always up to date. Clients should (where "should" is defined as the same as SHOULD in RFC 2119) cache videos locally to allow for videos to be served on demand in case of inability to reach the original video link (which is probably hosted on YouTube or something).

3. Test plan

We intend on writing the code correctly the first time.

4. "Challenges", or: why our system is better

4.1. **The 50% attack.** One of the technological challenges that plagues Bitcoin is the 50% attack: if one entity controls over 50% of hashpower, then this entity is able to subvert the security of the blockchain by forcing double-spends through. For this sort of attack to work with DollarCoin, one entity must control over 50% of circulating US $1 notes (or, probabilistically speaking, a significant share thereof). This is surely impossible. Furthermore, the difficulty of mining bitcoin incentivizes individual participants to conglomerate together in mining collectives known as "mining pools" in order to mitigate the variance of the otherwise random process of mining.

As of this writing, the top three Bitcoin mining collectives (one of which is simply an organization with its own custom mining hardware) control over 50% of hashrate, which means the security of the Bitcoin blockchain can be compromised by only three actors. With DollarCoin, there is no randomness in generation. Its security is derived from $1 notes

accessible to every individual, eliminating the need for conglomeration and making it far less susceptible to this sort of attack.

4.2. **Orphan blocks.** DollarCoin has no difficulty adjustment scheme because dollar bills are the only totally useless piece of currency with an extant alternative (dollar coins). One might surmise that the lack of difficulty adjustment could create a problem, as numerous simultaneous contributions to the blockchain result in many orphan blocks (blocks that no other blocks build upon).

This is not a problem. High numbers of orphan blocks will increase the relative scarcity of DollarCoin and cause its value to go to the moon.

4.3. **Transaction commitment times.** If your merchant requires more confirmations of a transaction than are currently present in the blockchain, simply burn more dollar bills.

4.4. **Exchange from traditional currencies.** DollarCoin has a built-in exchange in which users essentially deposit their cash via smartphone. This is currently a one-way exchange, but in the future we imagine banks will be happy to exchange your DollarCoin for dollar coins.

4.5. **Symbol for the currency.** 

5. Launch

The genesis block will be mined at SIGBOVIK 2014.

Towards a Completely Autonomous Vehicle

Nicholas Fudala Chester Francis

February 25, 2014

Abstract

Much progress has been made in fully-autonomous road vehicles. Yet even the most advanced self-driving cars must rely on a human to manually select a destination, as well as monitor the system in a support role. So needy. To address these shortcomings, we propose a completely autonomous vehicle with two novel features: (1) the ability to predict and enforce its destination based on a passenger's Web activity, and (2) the trait of self-actualization and emotional autonomy based on Maslow's hierarchy of needs. In theoretical testing, unexpected and occasionally embarrassing results were observed.

1 Introduction

The field of road vehicle automation has gained much attention with high-profile prototypes from Google [1], Nissan [2], and others. Although vehicles automation of throttle, braking, and steering are often referred to as "autonomous vehicles", others have argued that the term is a misnomer, as these vehicles require a human to input a destination [3]. For many, destination selection and input is a frustrating and unnecessary task, particularly given the extensive location-desired data available for any user. Beyond destination-selection, today's fully-autonomous vehicles remain heavily reliant on human interaction. An operator must monitor the system in the event of failure, provide fuel, and perform basic maintenance. With an autonomous vehicle's delicate sensors and complex software, these duties can only be described as "high maintenance." The vehicle itself can be described as "needy" or "bitchy."

In addition to the National Highway Safety Administration (NHTSA) levels of road vehicle automation [4], we propose additional requirements based on Maslow's hierarchy of needs [5]. As shown in Table 1, they both have five levels, which is nice. In this paper, we propose a vehicle that meets the fourth level requirements of both NHTSA (full self-driving) and Maslow (self-actualization).

Level	NHTSA [4]	Maslow [5]
0	None	Physiological
1	Function-specific	Safety
2	Combined function	Love/belonging
3	Limited self-driving	Esteem
4	Full self-driving	Self-actualization

Table 1: Levels of autonomy.

Destination	Frequency
AVN Awards Show, Las Vegas, Nevada	57%
Cat Fanciers' Association World Championship Cat Show, Novi, Michigan	25%
Google, Mountain View, California	10%
Continuous 2-4 mile circumference of user's residence[1]	8%

[1] In search of discreet local singles looking to f*ck.

Table 2: Projected destination frequency based on analysis of user (authors') Web activity.

2 Automatic Destination Selection and Enforcement

We seek to remove this human decision-event step through our destination prediction algorithm, utilizing the most proprietary, embarrassing, and presumably truly revealing aspects of a user's financial, location-based service, and Internet browsing history, with weighted emphasis on web browser privacy mode activity. The details of the algorithm are both conceptual, nonexistent, and far too complicated for this audience. See the results in Table 2 using the authors' joint data as test cases.

As we could never claim to be completely perfect, we have programmed an override function that will allow the human to express their opinion of where they want to go, no matter how wrong and feeble-minded they may be. Time lost due to manually over-riding the vehicle's selection will be discussed in the appendix of a different, unrelated study.

The potential time savings from the Automatic Intuitive Destination Selection System (AIDSS) cannot be underestimated. Wait, I mean, should not be overestimated. Look, we estimate a savings of 3 seconds per person per year (3000000 $\mu s/per/yr$), so don't estimate any higher or lower than that. Using motorist value-of-time measurements of \$21.46/hour in 2005 [6], our algorithm should yield potential savings of \$0.017 per person per year.[2]

3 Emotional Autonomy

An important component of a completely autonomous vehicle is emotional autonomy and self-actualization. We believe that our completely autonomous vehicle, through advanced emotional independence, will in fact yield the most authentic human-vehicle relationship ever devised. While this is obviously completely and totally beneficial over the long-term, we have experienced many barriers to implementation during initial testing. For example, the vehicle seemed, for lack of a better term, rarely in the "mood" for refueling, regardless of actual fuel level status. Some progress was made through compliments, outright flattery, and promises of windshield wiper replacement. The vehicle also refused all but expensive, premium fuel, this is spite of manufacturer's recommendation of 87 octane rating, and the researchers' repeated insistence that premium fuel is totally a scam. As these arguments are clearly independent from fact, they can be classified as nothing other than

[2] In 2005. Who knows how much that could be worth now!

excessive emotion, insofar as an artificial agent may experience. The vehicle's occasional vague suggestions of its potential to "do better" and desire to backpack Europe confirms its strong emotional autonomy. As such, we predict that there could be some trivial, inconsequential initial complications during beta testing that we're sure to have figured out right after publication, or Valentine's Day.

Additional concerns have also arisen about whether some vehicles may become too attached to their occupants while others expressed apprehension that the vehicles may want to move on and see other, more interesting, attractive, and open-minded occupants with improved senses of humor. We are developing, through early testing, a system that allows the vehicle to go on a nice drive—alone—through the hills so that it can defragment and ensure a happy, healthy vehicle-occupant relationship (VOR).

4 Conclusions

Early market research suggests deep penetration levels[3] for AIDSS among all age groups with exceptionally high demand found in the lucrative male age 18-49 bracket. We foresee such great time savings per person per decision-event based on AIDSS DEPPA that the resulting VORs may become VORRs (vehicle-occupant-romantic-relationships) and, if widespread enough, the authors might finally stop being ostracized in their communities.

References

[1] Burkhard Bilger. Auto correct: Has the self-driving car at last arrived? *The New Yorker*, November 25, 2013.

[2] Alan Ohnsman. Nissan sets goal of introducing first self-driving cars by 2020. *Bloomberg*, August 27, 2013.

[3] Steven Shladover. Using vehicle automation on freeways. 93rd Annual Meeting of the Transportation Research Board, Washington, DC, January 2014.

[4] National Highway Traffic Safety Administration. Preliminary statement of policy concerning automated vehicles. Technical Report NHTSA 14-13, National Highway Traffic Safety Administration, Washington DC, May 2013.

[5] Abraham Maslow. A theory of human motivation. *Psychological Review*, 50(4), 1943.

[6] Kenneth A. Small, Clifford Winston, and Jia Yan. Uncovering the distribution of motorists' preferences for travel time and reliability. *Econometrica*, 73(4):1367–1382, July 2005.

[3] A real thing.

Track 88,888

Easier, Better, Faster, Stronger Research Methodologies

1. **Belief-Sustaining Inference**
   Alex Reinhart and Jerzy Wieczorek
   Keywords: uninformative likelihood, statistics, embarrassment, cognitive dissonance, inference

2. **Solutions to Ley Line Access in Occult Computing**
   Maija Mednieks
   Keywords: ley lines, occult computing, miskatonic university, installing linux on a dead badger, playing god

3. **Measuring Your $\mathbb{P}$-ness: Determining How Like a Probability Distribution a Given Mathematical Object Is**
   John T. Longwood
   Keywords: lowbrow humour, probability, statistics

4. **a clarification of terminology**
   David Renshaw
   Keywords: did you mean, search instead for, not to be confused with

5. **What, If Anything, is Epsilon?**
   Dr. Tom Murphy VII Ph.D.
   Keywords: computational archaeology, epsilon, very-small and medium-small numbers

BELIEF-SUSTAINING INFERENCE*

By Alex Reinhart and Jerzy Wieczorek

Department of Statistics, Carnegie Mellon University

Two major paradigms dominate modern statistics: frequentist inference, which uses a likelihood function to objectively draw inferences about the data; and Bayesian methods, which combine the likelihood function with a prior distribution representing the user's personal beliefs. Besides myriad philosophical disputes, neither method accurately describes how ordinary humans make inferences about data. Personal beliefs clearly color decision-making, contrary to the prescription of frequentism, but many closely-held beliefs do not meet the strict coherence requirements of Bayesian inference. To remedy this problem, we propose belief-sustaining (BS) inference, which makes no use of the data whatsoever, in order to satisfy what we call "the principle of least embarrassment." This is a much more accurate description of human behavior. We believe this method should replace Bayesian and frequentist inference for economic and public health reasons.

1. Introduction. Modern statistics is at a crossroads. Frequentist inference, the original foundation of statistical inference, is under attack from many angles due to the low quality of work making use of it (e.g. Ioannidis, 2005, 2008) and its perceived philosophical paradoxes (Meehl, 1967). Using the likelihood function, frequentist inference attempts to infer parameters from the data objectively and with no reference to personal beliefs or subjectivity, making it very scientifically appealing but practically error-prone.

On the other hand, Bayesian inference is presented as a comprehensive system for the updating of personal beliefs on the basis of data. A rational Bayesian holds a coherent system of beliefs and systematically updates them as new data arrives. Computational difficulties rendered this method impractical until computers became sufficiently powerful, and it is now a hot area of research in statistics despite controversy about the appropriateness of subjectivity in science.

However, we believe that neither paradigm accurately represents how humans reason about their beliefs. As demonstrated by Kahneman, Slovic and Tversky (1982), most people do not hold coherent beliefs or update them as a Bayesian should, and the mere existence of subjective prior beliefs rules out frequentism as an accurate model.

*The authors acknowledge the funding and support of the Department of Statistics. The views expressed are those of the authors and will probably upset the Department when they find out they paid for this research.

MSC 2010 subject classifications: 62A01

Keywords and phrases: uninformative likelihood, embarrassment, cognitive dissonance, minimax

We believe both approaches are misguided. Rather than attempting to minimize error, as in frequentism, or attempting to maintain a coherent set of subjective beliefs, most people attempt to *minimize embarrassment*.

This insight leads to a new class of estimators we refer to as *belief-sustaining inference*, or BS inference, which we shall discuss in the following sections. An important discovery is that embarrassment is minimized by ignoring the data altogether. We also show that this approach has important public health benefits.

2. Quantifying embarrassment. The principle of minimum embarrassment may be explained in terms of the theory of cognitive dissonance (Festinger, Riecken and Schachter, 1956). The arrival of new data causes a conflict in the mind of the scientist: he would like to believe he is a rational, intelligent person who holds correct prior beliefs, but the data suggests he is wrong. This dissonance causes psychological distress and can only be resolved by jettisoning one of the contradictory beliefs, such as one's self-esteem. Overwhelming or unimpeachable evidence can thus cause severe embarrassment and psychological breakdown.

To minimize embarrassment, it is first necessary to define the mathematical concept of embarrassment in terms of the change in a personal prior after data is collected.

Definition 2.1. *Let X be a random variable distributed according to some distribution function $f(x;\theta)$, where θ is an unknown parameter, $\pi(\theta)$ a personal belief about that parameter, and $\pi(\theta|x)$ the estimate based on the data. The* ***embarrassment*** *E is the distance between $\pi(\theta)$ and $\pi(\theta|x)$, as measured by the Kullback-Leibler divergence:*

$$E = \int \log\left(\frac{\pi(\theta)}{\pi(\theta|x)}\right)\pi(\theta)\,d\theta. \tag{1}$$

In Bayesian inference, the data (in the form of the likelihood $p(x|\theta)$) is combined with the prior $\pi(\theta)$ to produce a new best estimate of the parameter, using Bayes' famous theorem:

$$\pi(\theta|x) = \frac{p(x|\theta)\,\pi(\theta)}{\int p(x|\theta)\,\pi(\theta)\,d\theta}. \tag{2}$$

Many approaches are taken to choose the appropriate prior distribution, and a great deal of literature deals with the elicitation of priors from subject-matter experts. Other work attempts to eliminate subjectivity by choosing an "uninformative" or "flat" prior (*e.g.* a constant function) which places no special importance on any specific value of the parameter, letting the data decide instead.

In frequentist inference, the posterior estimate does not depend on $\pi(\theta)$, and can be a point or interval estimate solely based on $p(x|\theta)$ (*e.g.* the value of θ which

maximizes $p(x|\theta)$). The estimate usually has an asymptotic normal distribution. But if we wish to minimize embarrassment, both approaches are wrong-headed, as demonstrated by the following theorem.

THEOREM 2.2. *In Bayesian inference, the embarrassment E is minimized by choosing an* uninformative likelihood*, also known as a* flat *likelihood.*

PROOF. Let $p(x|\theta) = 1$. ($p(x|\theta)$ may differ on zero-measure sets in θ, with respect to the Lebesgue measure.) Using eq. (2) and the fact that the probability density $\pi(\theta)$ integrates to 1, we find that $\pi(\theta|x) = \pi(\theta)$.

Substituting into eq. (1), we determine that

$$E = \int \log\left(\frac{\pi(\theta)}{\pi(\theta)}\right)\pi(\theta)\,d\theta = 0. \tag{3}$$

The Kullback-Leibler divergence is always nonnegative, so this embarrassment is minimal. We leave the proof that this solution is unique as an exercise for the reader. □

That is, we should not allow the data to place special importance on any specific value of the parameter, as the data will not feel obligated to support the value most beneficial to the scientist. It may even prove embarrassing, and this must not be allowed. Belief-sustaining inference requires that we ignore the data instead of taking this risk.

(Some readers may prefer to think of this in the minimax framework. Instead of aiming for the minimax risk, we aim for the minimax embarrassment. The maximum embarrassment would be a final parameter value entirely contrary to the prior, and this embarrassment is minimized by never allowing the parameter estimates to deviate from the prior.)

3. Belief-sustaining inference. Ordinarily, we would use this section to discuss the benefits of belief-sustaining inference and properties of the data-free estimator. However, any attempt to derive these properties could prove embarrassing.

Nonetheless, we will point out several important features of belief-sustaining inference. Because the belief-sustaining estimator has zero variance, it gives narrower confidence intervals than any other technique, and similarly is maximally efficient. Statistical power calculations, usually so complex as to merit entire textbooks on the subject, are made simple: the most cost-effective number of samples is always zero. Computer clusters previously wasted obtaining Bayesian Markov Chain Monte Carlo estimates can be put to more productive uses, for BS inference is computationally efficient.

We must also recognize an important public health benefit of the method of least embarrassment. Bayesian and frequentist inference would have us constantly change our beliefs, subjecting us to cognitive dissonance and causing a great deal of stress. This stress can lead to heart attacks, strokes, and other unpleasant outcomes: according to the American Institute of Stress, stress costs Americans over $300 billion annually in medical, legal and productivity costs. Hence frequentist and Bayesian inference impose a $1,000 per person tax on Americans who are already economically struggling, while the adoption of embarrassment-free inference would give an immediate 2% boost to the recovering American economy.

Considerable evidence thus suggests that evidence should be ignored altogether.

4. Conclusions. We have demonstrated that rational actors following the principle of least embarrassment will rightly ignore new data, for fear it might cause an embarrassing change of position. Ample sociological evidence demonstrates the accuracy of this model; for a particularly high-profile series of experiments, watch any Presidential debate or political talk show. The foundations of Bayesian and frequentist inference are hence falsified, and public health concerns suggest they should be banned entirely.

This work has impact on many questions of current interest. For example, scientific publishing is currently under attack by open-access advocates who would have us make publicly-funded research freely available to anyone who wishes to read it, regardless of the risk of embarrassment and cognitive dissonance to the unsophisticated reader. While professional scientists have spent years developing defensive mechanisms to protect themselves from embarrassing results, the unsuspecting reader might be unintentionally exposed to an idea contradictory to their naïve and unscientific beliefs. Our developments in embarrassment theory clearly demonstrate the foolishness of such proposals.

Additionally, BS inference has important applications in many fields of study. Economists, for instance, will be heartened by the requirement to never test theories against empirical data. Previously computationally-intractable problems in other fields are rendered trivial.

Further research is merited on several questions. For example, is it possible for the weight of evidence to be so strong that *not* changing one's opinion is *more* embarrassing? Researchers are encouraged to send us their results, although we will of course ignore them.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their undoubtedly insightful comments, which of course we did not read.

REFERENCES

FESTINGER, L., RIECKEN, H. and SCHACHTER, S. (1956). *When Prophecy Fails: A Social and Psychological Study of a Modern Group That Predicted the Destruction of the World*. Harper-Torchbooks.

IOANNIDIS, J. P. A. (2005). Why Most Published Research Findings Are False. *PLOS Medicine* **2** e124.

IOANNIDIS, J. P. A. (2008). Why Most Discovered True Associations Are Inflated. *Epidemiology* **19** 640–648.

KAHNEMAN, D., SLOVIC, P. and TVERSKY, A. (1982). *Judgment under Uncertainty: Heuristics and Biases*. Cambridge University Press.

MEEHL, P. E. (1967). Theory-testing in psychology and physics: A methodological paradox. *Philosophy of Science* **34** 103–115.

E-MAIL: areinhar@stat.cmu.edu

Solutions to Ley Line Access in Occult Computing

Maija Mednieks

Carnegie Mellon Department of Occult Computing
mmednieks@gmail.com

Abstract

As the field of occult computing has developed, practitioners have run up against problems of power consumption in the local aether. In the initial stages of this field, programs were simple enough to run off of local aetherial energy; however, anything beyond the simplest computations require more mystical energy to run efficiently. While it is possible to construct a program to perform every conjuration sequentially, this increases running time exponentially,For better performance, ceremonies should be performed in more energized areas, that being ley lines and intersections of ley lines. The problem arises that many established centers of occult computing do not have easy access to ley lines. In retrospect this was a huge oversight. This paper explores three possible solutions to this problem.

I. Lyne: Your friend with a Dead Badger

The simplest solution, other than travelling to a ley line oneself, has been to locate someone claiming to be experienced in incantations, and request that they execute your program, and compensate them for their time and any materials required. This has been plagued by frauds and charlatans, who either will not perform the ritual required, or will use sub-standard components. This lack of regulation has limited the number of new enthusiasts of occult computing, either due to stories of these conmen, or being taken in by one. Lyne, a startup founded at one of the major intersections, Salt Lake City, has decided to tackle the lack of accountability in outsourcing incantations. One simply downloads their app, and send the specifications of the rite they want performed to one of Lyne's experienced occult computing practitioners. To ensure a program is cast properly, two crowdsourced thaumaturges are on hand to oversee chanting, and anyone intentionally botching a rite is promptly removed from their list of employees. This is an excellent solution for anyone starting out in the field, as it allows for reliable, relatively inexpensive executions of occult programs. The price increases sharply as the length of a chant extends past seventy-seven minutes, requires more than four dead languages, involves or precisely three animal sacrifices, so for more complicated ceremonies this is not an ideal solution.

II. Occult Supercomputing

For those working with an academic institution, University of Amsterdam is opening an occult supercomputing center. As the first installation of its kind, it is located at a major European ley line nexus, and will have new, state of the art computers with aethernet components fully integrated with the computer. Its opening next week will show how the university is planning to handle components that are consumed during computational rituals will be handled, particularly during non-business hours; however if this is seen as a success, similar centers are slated to open in North Carolina, Hawaii, Australia, and Siberia. Accessing the power contained in one of these centers is limited to members of registered occult computing circles, and so is not a good solution to those who are working in secret, or are currently

banished from the AOCM.[1]

III. Straight Up Playing God

Finally, researchers at Miskatonic University are exploring ley line creation. This appears promising as it would make access a simple matter, even for hobbyists. Unfortunately this is an entirely new undertaking, and after the disappearance of Bolton, the neighboring factory town, many are protesting further research in this matter. This has not deterred Miskatonic's researchers, if anything they are frantically working to create an artificial ley line before legal action is taken against their occult computing department. It is unclear what effects a manmade ley line will have on the global, or local, aetherial environment, as the precise interactions of occult computation and ley lines is still under investigation, and so it will take observations of the ley line once created to determine the side effects of toying with forces we can barely begin to comprehend.

[1] The University of Amsterdam subscribes to the idea that time exists, even if this conflicts with indivual personal beliefs, or reality itself.

SIGBOVIK 2014 Paper Review

Paper 13: Solutions to Ley Line Access in Occult

Emily Forney, Food Acquisition Department
Rating: 666 (weak accept)
Confidence: 3/4

This paper offers three fairly good solutions to the problem of not much key lime access in occult computing. I underderstand that accessing key lime is not easy as occult computing becomes more demanding of resources. But the paper has some good ideas, given the difficulty of the situation. It suggests working with a startup which puts you in contact with a key lime baker (hopefully no dead badger is called for in the recipe), collaborating with a university, or in fact making your own key lime. However, the key lime is an ancient and well-kept recipe, so making your own seems like the most dangerous option.

Measuring Your $\mathbb{P}$-ness: determining how like a probability distribution a given mathematical object is

John T. Longwood*
Unemploy(ed|able)

Abstract

It is common in certain machine learning scenarios to hallucinate a probability distribution from data using ad-hoc methods. These distributions often do not actually need to be true probability distributions but, rather, can simply approximate them to a degree suitable for the application at hand. In this paper, I introduce a measure to quantify the probability-likeness (hereafter, "$\mathbb{P}$-ness") of such ad-hoc objects. Among other advantages, being able to directly measure an object's $\mathbb{P}$-ness should allow for less guesswork and speculation in procedure design.

CR Categories: ____ [Blank]: Blankings—Blank

Keywords: lowbrow humour, probability, statistics, penis, dong, stiffy, dick, tadger, prick, willy, john thomas, one-eyed trouser snake, piece of pork, wife's best friend, percy, cock, oblique monty python references

1 Introduction

Probability distributions describe how random variables vary. To wit, we say random variable X takes values in S distributed in accord with distribution $\mathcal{D}$ ("$X \sim \mathcal{D}$") if the probability that X falls into a given subset $s \subseteq S$ is given by $\mathcal{D}(s)$[1].

$$X \sim \mathcal{D} \Rightarrow \mathbb{P}(X \in s) = \mathcal{D}(s) \tag{1}$$

Of course, for this equation to make sense, $\mathcal{D} : 2^S \to \mathbb{R}$ must obey some simple principles:

2. When applied to an empty set, D should return zero, and when applied to the whole of S, D should return 1 – because any random variable must take some value (and can't take no value)[2].

3. D should never produce a value less than zero or larger than one – those aren't probabilities[3].

4. When applied to disjoint sets, the sum of the values D produces should be equivalent to that produced by applying D to their union – since the probability of disjoint outcomes sums[4].

Or, in plain English:

$$\mathcal{D}(\emptyset) = 0, \mathcal{D}(S) = 1 \tag{2}$$

$$\forall s \subseteq 2^S \quad : \quad \mathcal{D}(s) \in [0, 1] \tag{3}$$

$$\forall s \subseteq 2^S, t \subseteq 2^{S-s} \quad : \quad \mathcal{D}(s \cup t) = \mathcal{D}(s) + \mathcal{D}(t) \tag{4}$$

2 Background

The strict conditions on probability distributions are as old as the science of gambling [Vegas 1502]. However, only recently has it become faddish to relax these conditions. In fact, my work appears concurrently with another, divergent derivation of $\mathbb{P}$-ness [Lax and Silly 2014]. Given their weaker conditions, I suggest that Lax and Silly's definition be styled "soft $\mathbb{P}$-ness", in contrast to my "hard $\mathbb{P}$-ness".

However, for the remainder of this paper I will simply use the term "$\mathbb{P}$-ness", as I think my hard $\mathbb{P}$-ness is really the only thing worth talking about.

3 Showing Off Our $\mathbb{P}$-ness

My notion of probability-likeness is arrived at by transforming each of the terms of the traditional definition presented above (at the start of the paper (in the introduction (Section 1))) into a term in an overall energy function. These constituent terms are each normalized to range from 0-3000, allowing the overall power level to reach 9000 (see [Kajetokun 2006] for a fuller treatment).

Thus, my $\mathbb{P}$-ness – designated σ[5] – has three main attributes:

$$\sigma(\mathcal{D}) \equiv \sigma_L(\mathcal{D}) + \sigma_G(\mathcal{D}) + \sigma_R(\mathcal{D}) \tag{5}$$

These constituent properties – Length, Girth, and Rigidity – are described in more detail in the following subsections.

*ix@tchow.com

[1] In this paper, I will conflate the notion of a distribution and a probability density/mass function. If this worries you, maybe you shouldn't be reading the proceedings SIGBOVIK. Seriously, can't a person get a break here?

[2] Discussion of *nullable probability* and the "$X \sim \mathcal{D}$ `throw`" syntax are both beyond the scope of this document.

[3] Discussion of *exceptional (im)probabilitiy* is beyond the scope of this document.

[4] Discussion of *discordant probability* and the field of *over-unity statistics* is beyond the scope of this document.

[5] "... for obvious anatomical reasons that Waterhouse finds amusing at this stage of his emotional development" – Neal Stephenson.

3.1 Length

Length measures how much D accords with principle (4) of distributions by penalizing[6] any failure in summation:

$$\sigma_L(\mathcal{D}) \equiv \frac{3000}{1 + \sum_{s \in 2^S, t \in 2^{S-s}} |\mathcal{D}(s) + \mathcal{D}(t) - \mathcal{D}(s \cup t)|} \tag{6}$$

3.2 Girth

My $\mathbb{P}$-ness's girth is structured to reflect principle (3):

$$\sigma_G(\mathcal{D}) \equiv \max\left(0, 3000 - \sum_s - \min(0, \mathcal{D}(s)) + \max(0, \mathcal{D}(s) - 1)\right) \tag{7}$$

(With apologies to the readers for any discomfort caused by my substantial girth – it has been difficult finding LaTeX to appropriately contain it.)

3.3 Rigidity

Finally, rigidity reflects adherence to the most stringent principle – (2):

$$\sigma_R(\mathcal{D}) \equiv \begin{cases} 3000 & \text{if } \mathcal{D} \text{ follows principle (2)} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

4 Examples

Having defined our measure, we think it is instructive to more closely examine the $\mathbb{P}$-nesses of several example (pseudo-)distributions.

4.1 Example: An actual, totally 100% real, probability distribution

As a first example, let us introduce this friendly little distribution:

$$\mathbb{P}(X = x) \equiv \begin{cases} 1 & \text{if } x = 1 \\ 0 & \text{otherwise} \end{cases}$$

Our friend is a simple distribution indeed, and it is clear that the true properties that hold true of a true distribution truely hold of it in truth.

We can represent this distribution by $\mathcal{U}$, tabulated over $2^{\{0,1\}}$ as follows:

$$\begin{aligned} \mathcal{U}(\emptyset) &\equiv 0 \\ \mathcal{U}(\{0\}) &\equiv 0 \\ \mathcal{U}(\{1\}) &\equiv 1 \\ \mathcal{U}(\{0,1\}) &\equiv 1 \end{aligned}$$

[6] Hur hur hur. "Penal." Hur hur hur hur.

Let's get to know our friend a bit better by measuring the size of their $\mathbb{P}$-ness:

$$\sigma_L(\mathcal{U}) = \frac{3000}{1+0} = 3000$$

$$\sigma_G(\mathcal{U}) = 3000 - 0 = 3000$$

$$\sigma_R(\mathcal{U}) = 3000$$

Therefore:

$$\sigma(\mathcal{U}) = 3000 + 3000 + 3000 = 9000$$

As expected, our friend has an impressive $\mathbb{P}$-ness. I'm sure we could do quite a lot with it.

4.2 Example: A Pseudo-Distribution

Now, consider the following somewhat flawed distribution (also over $\{0, 1\}$):

$$\begin{aligned} \mathcal{F}(\emptyset) &\equiv 0 \\ \mathcal{F}(\{0\}) &\equiv 0 \\ \mathcal{F}(\{1\}) &\equiv 0.5 \\ \mathcal{F}(\{0,1\}) &\equiv -1.0 \end{aligned}$$

It disobeys every rule of probability distributions; but what does it's $\mathbb{P}$-ness look like?

$$\sigma_L(\mathcal{F}) = \frac{3000}{1 + 1.5 + 1.5} = 750$$

$$\sigma_G(\mathcal{F}) = 3000 - 1 = 2999$$

$$\sigma_R(\mathcal{F}) = 0$$

Thus:

$$\sigma(\mathcal{F}) = 750 + 2999 + 0 \approx 0 + O(1)$$

This pseudo-distribution's $\mathbb{P}$-ness appears tragically flaccid.

4.3 Example: Time-varying distribution

My measure of probability-likeness is also suitable for the summarization of the changes in pseudo-distributions over time. For instance, in Figure 1 I show a picture of the $\mathbb{P}$-ness of a distribution during a recent evening's experimentation.

As you can see, the probability-likeness of the pseudo-distribution being measured reaches a climax around $t = 3.0$ after which its $\mathbb{P}$-ness slowly deflates – during what I term the "refractory period" – before building to a second climax around $t = 22.0$ and deflating to quiescence.

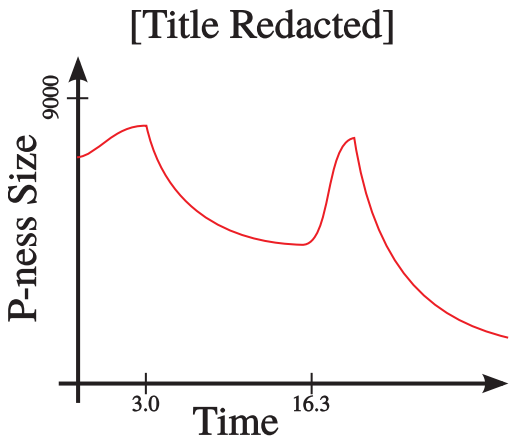


Figure 1: *"Clearly linear"*[7]

5 ℙ-ness Extensions

Intuitively, it seems like it shouldn't so much be the size of the ℙ-ness that matters, but – rather – how you intend to use the distribution being measured.

However, if situations arise in which this is not the case, it might be worthwhile for researchers to investigate normalization, canonicalization, and other methods of ℙ-ness enlargement.

It is also worth noting that my ℙ-ness is a tool for practical – rather than theoretical – statistics. Thus, a distribution with a large ℙ-ness may not necessarily be more fun to reason about, e.g., in bed.

6 Conclusions

In the weeks and months and days and years and decades and fortnights and minutes and seconds leading up to the publication of this paper, I have found immense pleasure in playing with my notion of probability-likeness. I hope that you will derive as much enjoyment when inserting it into your own research.

Acknowledgements

I'd like to thank my close collaborators, who – having had the opportunity to play with various versions of my ℙ-ness in meetings, at presentations, and during lunches – urged me to display it to the rest of the research community. Without their hands-on attitude, my ℙ-ness would not be nearly the measure it is today.

[7] Yep, that's an "in" joke. I figure it's actually raising the level of the humor in this paper, as pretty much everything else is either a dick joke or self-deprecation.

References

KAJETOKUN, 2006. 9000!! NINE THOUSAAAAANDD! https://www.youtube.com/watch?v=TBtpyeLxVkI.

LAX, L., AND SILLY, S. 2014. A weak notion of the probability-like-ness of mathematical objects. *Proceedings of ACH SIGBOVIK*. (fake).

VEGAS, L., 1502. Gambling. Nevada.

a clarification of terminology

David Renshaw

Figure 1: flow control

Figure 2: control floe

SIGBOVIK 2014 Paper Review

Paper 20: a clarification of terminology

Henrik Svensson, Svalbard University
Rating: 3 (accept)
Confidence: 2/4

This paper addresses an important point of terminological confusion. However, it ignores the issue of *floe control*, the global reduction in ice championed by climate change deniers.

What, if anything, is epsilon?

Dr. Tom Murphy VII Ph.D.*

1 April 2014

Abstract

We present a sample of the values of the programming constant `epsilon` as found on the internet, for several different programming languages and with a variety of visualizations.

Keywords: computational archaeology, epsilon, very-small and medium-small numbers

Introduction

Epsilon, the all-spelled-out version of ϵ, although properly "epsilon" because it's the *lowercase* Greek letter, though it's not like I'm going to start my paper with a lowercase letter even if it's technically correct, since I like to wait at least until the second or third letter of the paper before the reader starts doubting that I can write or spell or have shift-keys on my keyboard, anyway epsilon is a mathematical symbol denoting a *very small number*.

In mathematics, ϵ usually refers to the ϵ–δ formulation of limits. This is pretty simple and a reminder appears in Figure 1. This paper is not about that kind of math.

In computing, ϵ is used in a much more general sense to just mean some small number or error bound. For example, two numbers are often considered equal if their absolute difference is less than ϵ.

In IEEE-754 floating-point [2], the standard way that computers represent "real numbers," there is a specific formal value called "machine epsilon", or "unit roundoff". It is the maximum (relative) amount of error from a single rounding operation. This number is useful if you want to do numerical programming and be careful about what you're doing.

Most programmers find this subject too tedious and simply pick a number that seems pretty small. Thus in practice, ϵ is an application-specific choice. Of well-known "constants," ϵ may be the least agreed-upon, with values seen in the wild spanning more than *300 orders of magnitude*. This paper explores the practical values of ϵ in real software.

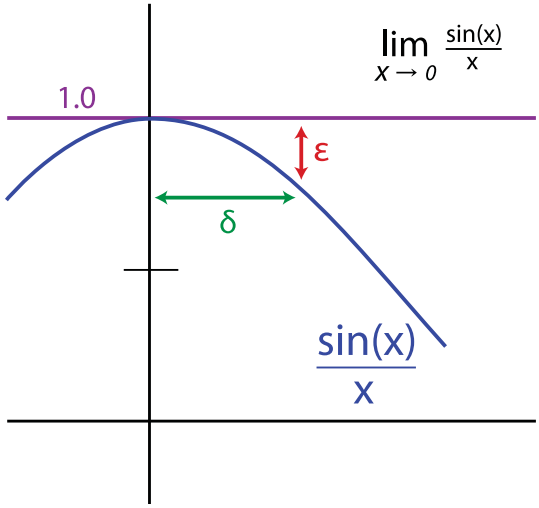


Figure 1: The curvy line is the function $\frac{\sin x}{x}$, which is undefined at 0. However, its limit as x approaches 0 is 1. The ϵ–δ formulation of the limit is this: For any positive choice of ϵ, there exists some δ such that $\frac{\sin(0+\delta)}{0+\delta}$ is less than $1+\epsilon$. In other words, for an arbirarily small error (your choice), I can produce a delta from 0 (my choice) that brings the result within the error of the limit. This has nothing to do with the subject of the paper.

1 Methodology

Github. Github is the hub that contains all gits, approximately 10 million of them, as of the beginning of

2014. The site has search functionality, which "allowed" me to scrape one hundred pages of results for queries like `"const double epsilon ="` for various languages, as long as I didn't do it too fast. I scraped the programming languages C, C++, C♯, JavaScript, and Objective C. Each language has its own idiosyncracies about how constants are defined, so I used one (or several) appropriate to each language. For example, in JavaScript, I looked for `"var epsilon ="`. From these HTML files I extracted all of the right-hand-side expressions, manually excluded the ones that could not be evaluated (for example because they depended on other symbols; see Figure 2 for some examples), and then computed the actual values for the rest. The source code to do the scraping, extract the expressions, and tally the results is available online.[1]

```
0.5/ELEC_REST_ENERGY
alpha/beta
4 / MULT32
exact_epsilon(true)
fmass_Epsilon * EPS_EXTRA
((Lj_Parameters*) parameters)->
scalar_traits<
EpsArray[prec]
hfwfn_->
fl.net_.opt_.epsilon
Tolerance
```

Figure 2: Other uninterpretable values of `const double epsilon` in C. Who knows what these are supposed to be?

SPEC benchmarks. Did you know that the SPEC benchmarks [1] cost $800? Like they literally expect me to pay them money to download the source code so that I could grep for `const double epsilon` or test my compiler out on that. Many are even based on open-source software like Sphinx and POV-Ray. Ridiculous. I refuse. Values of epsilon for the SPEC benchmarks do not appear in Figure 3.

2 Results

The results of the analysis appear in several figures which are interspersed haphazardly with this text. Each figure presents the data in a different way, since this diversity in presentation should maximize the chance that

[1] In the Subversion repository at: `https://sourceforge.net/p/tom7misc/svn/HEAD/tree/trunk/epsilon/`

Figure 3: $800? Fuck that!

one of the charts makes sense to you. The C programming language has two numeric types that could reasonably be used to represent ϵ: `float` and `double`. The results for `double` appear in Figure 4 and for `float` in Figure 5. C++ has those same two types, but I decided arbitrarily to only look at `double`, which is in Figure 6. Programmers in C♯ are very creative; their results are presented in Figure 7. Objective C proved unpopular for use of ϵ, its sparse data are in Figure 8. Finally, the ineffable JavaScript has its results in Figure 9.

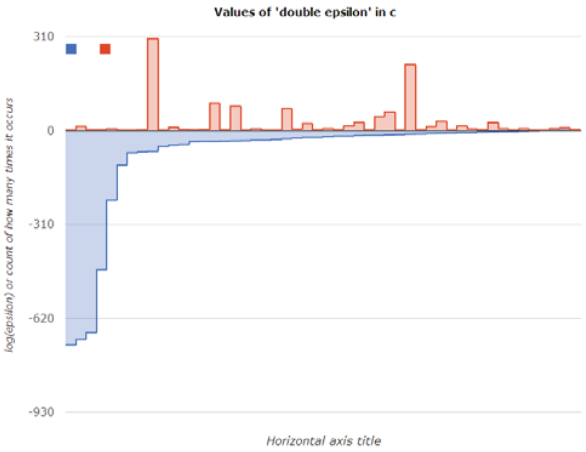


Figure 4: Values for `const double epsilon` in the C programming language. In this chart, the blue (lower) bars are the distinct values of epsilon seen. The vertically aligned red (upper) bar is its count. Epsilon values are plotted on a logarithmic scale, where the minimum observed value $\log(-708)$ is 303×10^{-308}, and the largest $\log(1.609)$ is 5. Notes: One programmer used the value `-1e10`, which is -10,000,000,000, probably meaning `1e-10`. This value was excluded because it has no real logarithm.

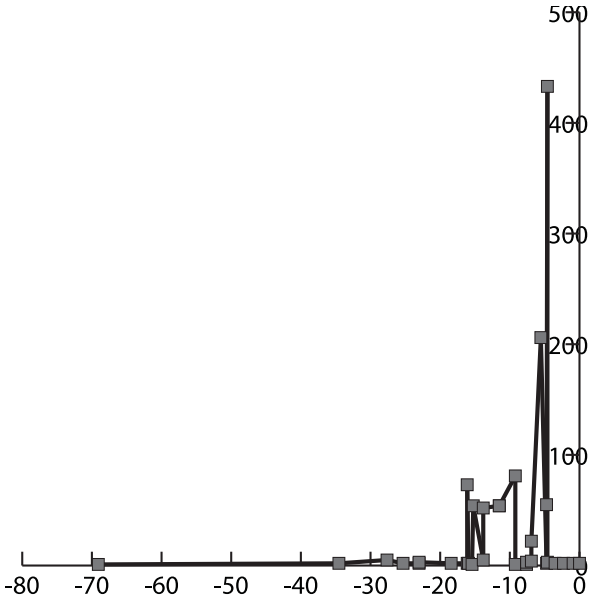


Figure 5: Values for `const float epsilon` in the C programming language. An x–y scatter plot where the y coordinate is the count of the number of times that specific value occurred, and the x coordinate is the log of the value. Values take on a less extreme range than with type `double`, naturally, ranging "only" 31 orders of magnitude from 3.9×10^{-31} to 0.

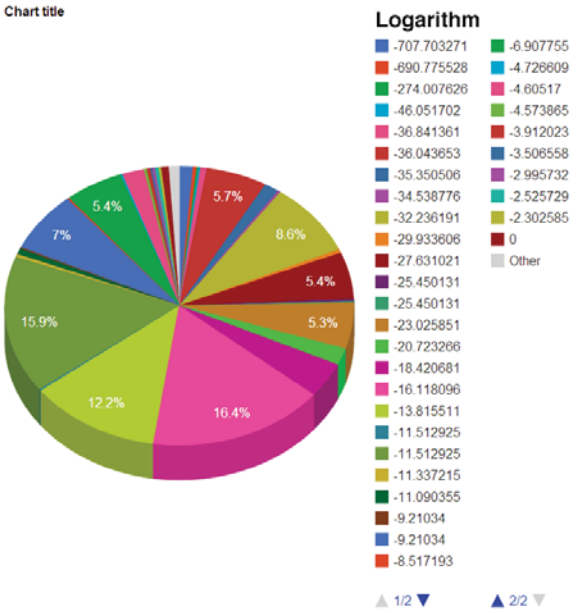


Figure 6: Values for `const double epsilon` and `constexpr double epsilon` in the C++ programming language. The `constexpr` qualifier, a new feature of C++11, is used very rarely (less than 1% of the time). Notes: Five times, the programmer used 0.0 for epsilon, which is possibly the only unjustifiable value. The value `pow(10,-13)` is annotated in German "Genauigkeitsziel bei der Nullstellensuche," or "Accuracy goal in the search for zeros."

References

[1] John L. Henning. Spec cpu2006 benchmark descriptions. *SIGARCH Comput. Archit. News*, 34(4):1–17, September 2006.

[2] IEEE Task P754. *ANSI/IEEE 754-1985, Standard for Binary Floating-Point Arithmetic*. IEEE, New York, NY, USA, August 1985. Revised 1990.

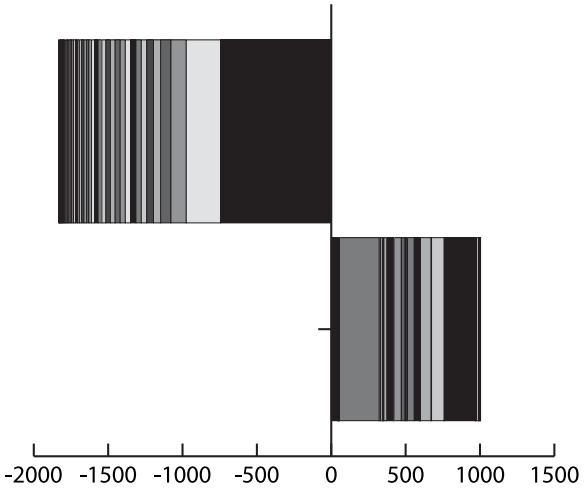


Figure 7: Values for `const double epsilon` in the C♯ programming language. In this chart, the left edge of greyscale blocks on the left-facing flag represent the cumulative value of the logarithm of all values of epsilon seen so far. The right edge of the block with the exact same greyscale value on the bottom-right flag is the cumulative count of all definitions of epsilon so far. It is easy to see that there are 1,000 total definitions, as expected. Notes: One programmer defined epsilon as `1.0-18`, that is, -17, probably meaning `1.0e-18`. Four used literal `0` for epsilon. Another defined epsilon as `1 / 100000`, which uses integer division and results again in 0. Owing to the strength and creativity of C♯ developers, here we saw our largest value of epsilon so far, `700`, and the declaration with the most significant digits: `0.00000000000000022204460492503130808472633361816406250`.

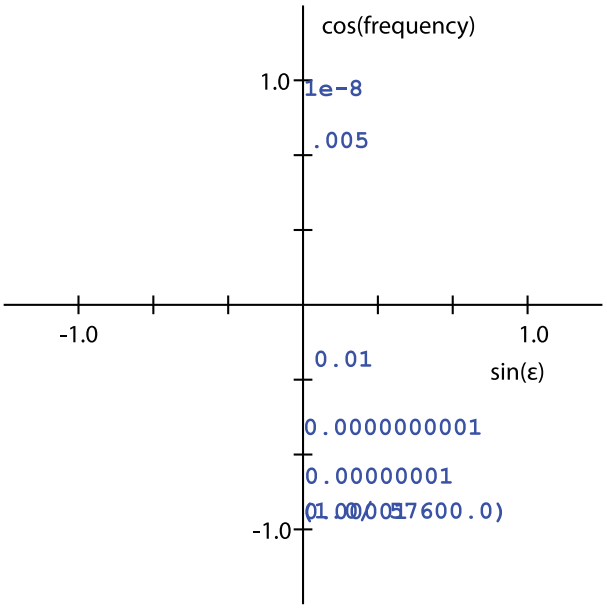


Figure 8: Results for the language Objective C for `const double epsilon`. Perhaps because Objective C programmers do not use this idiom for defining constants, or because it is only used to make iPhone games where epsilon is not a concept of interest, there were only 8 distinct values of epsilon observed. This allows us to present the data completely in the chart. The x axis is $\sin(\epsilon)$ and the y axis is $\cos(f)$ where f is the total number of times that the given value occurred in the code.

Figure 9: Finally, JavaScript, the official language of the Internet. In this "radar" plot, the data are arranged around a circle, as tuples of $\langle \log(\epsilon), \epsilon, f \rangle$ where f is the frequency of the value being observed. Numbers in boxes ascend along the 0 deg axis, displaying each of the multiples of 5 between 0 and 100, inclusive. Boxes stack so that only part of the number is visible, but you know what's under there if you've looked at numbers before. A spider-web-like network of interlacing lines ascribe some additional meaning to some of the points on the clock face. The smallest nonzero value observed was 10^{-32}. Notes: JavaScript programmers have the highest tolerance for error of all languages tested, with over 1,000 using epsilon of 0.5 or higher. One programmer used $\epsilon = 1024$, and another 6,000,000!

Track 888,888

Innovations in Natural and Unnatural Languages

1. **IRS-1040-Based Computing: The Untapped Computational Power of Bureaucracy**
   William Gunther and Brian Kell
   Keywords: unconventional computation, taxes, compilers, bureaucracy

2. **Towards Fooling and Foiling the NSA: Surveying the State of Steganographic Programming Languages**
   Wolfgang Richter and Debjani Biswas
   Keywords: steganography, esoteric programming language, esoteric, programming languages, obfuscation, cryptography, shit

3. **Pikachu, Domosaur, and other Monolexical Languages**
   Sarah Allen, Jesse Dodge, and Domosaur Domosaur
   Keywords: natural language processing, regular languages, translation

4. **Unit-Test-Based Programming**
   Miguel Á. Lechón
   Keywords: unit test, automatic stubbornness, holy grail

IRS-1040–based computing: The untapped computational power of bureaucracy

William Gunther Brian Kell

April 1, 2014

Abstract

As the computational needs of science and industry increase, it is necessary to seek out and develop new forms of computation. Current research is actively ongoing in such fields as molecular computing, biological computation, quantum computing, atomtronics, and optical computing.

However, there is a potential computational platform that so far has not been fully appreciated. A folklore theorem states that there are but two certainties in life: death and taxes. Given this guarantee of perpetuity, it is surprising that the tax system has not been investigated as a means for computation. Companies such as H&R Block, Jackson Hewitt, and Liberty Tax Service employ over 165,000 tax preparers, but the work they do is primarily done between February and mid-April. For the rest of the year, these tax preparers collect dust in various closets all over the country, where they are largely forgotten. We see this as a clear waste of computational resources that can and should be put to a useful purpose.

To this end, we propose the development of an IRS-1040–based architecture as a computational platform. Fortunately, the complexity of the United States tax code provides a ready-made foundation for computation. Tax preparers execute programs that are written in a rich machine language, organized as tax forms. We show that this system is capable of expressing any computable function, which means that all computational problems of interest to science and industry can be encoded as tax forms. Tax preparers can then be kept working year round to the benefit of society.

In order to facilitate this goal, we have implemented a C compiler for this platform, called irscc. We will demonstrate the use of this compiler and the effectiveness of this new computational platform. We hope that this will inspire further research in this promising area.

SIGBOVIK 2014 Paper Review

Paper 14: IRS-1040-based computing: The untap

Alvin P. Worthington III, Internal Revenue Service
Rating: 1 **(reject)**
Confidence: 4**/4**

This work is promising, but IRS-1040 has a number of shortcomings as a computation platform which I'm worried cannot be overcome. A notable one is that one would seem to be unable to compile 64-bit programs, as 2^{64} is considerably larger than the United States GDP, and 64-bit ints would overflow the computational ability of tax preparers. Despite these concerns, I would recommend the paper be accepted were it not for the following paperwork errors in the submission:

1. Title is not properly in Title Case [1].
2. Title is too long (as is made clear in the header of this review)
3. Paper must have five or six keywords, and has only four.
4. Abstract contains 303 words, violating strict limit of 300.

I recommend that this submission be published as an abstract only, both because of the above concerns and because that is all that was submitted.

[1] http://blog.apastyle.org/apastyle/2012/03/title-case-and-sentence-case-capitalization-in-apa-style.html

Towards Fooling and Foiling the NSA: Surveying the State of *Steganographic* Programming Languages

Wolfgang Richter and Debjani Biswas
Carnegie Mellon University

Abstract

Trust is non-existent in a post-Snowden world where our worst fears have become reality. The Internet is now a surveillance state and every action taken with a digital device is likely spied upon in some form. Nation-states are deliberately tampering with and weakening cryptographic systems making eavesdropping trivial. Eve is very real, and her resources are virtually unlimited. With cryptosystems currently under attack, what can normal computer scientists do to hide their activities? They must turn to a close cousin of cryptography: steganography. In this paper, we explore programming languages which have steganographic value.

1 Introduction

If you are a computer scientist and you wish to transmit executable information in a secret fashion, cryptography is the natural choice. We have strong asymmetric- and symmetric-key cryptosystems [20, 23]. We even have theoretically unbreakable cryptosystems [22]. However, powerful nation-states are actively working to reduce the strength of practical implementations of cryptography [17, 18, 21, 26]. Unfortunately, attackers always win given enough time and resources. With the revelations of Edward Snowden [24], what can we do to securely share our executable messages?

If we can not trust cryptography, we must use a different methodology altogether. In this paper we explore the potential of steganographically [25] communicating executable content. Of course, we could use standard steganographic techniques such as embedding inside an image. But, this software is not widely available and furthermore it could be tampered with to reduce its effectiveness. No, instead we search for a programming language solution. We desire a programming language that is *inherently* steganographic.

Luckily, programming languages research literature is rife with languages suitable for steganography. The technical term they use for such languages is 'esoteric' [7]. In this paper we survey esoteric languages searching for a practical choice facilitating the steganographic exchange of executable information. Along the way, we discover the surprising result that a language with *high steganographic value* is already deployed on billions of devices around the world.

2 Steganographic Value

In order to rank potential programming languages for their steganographic value, we need a quantitative measure to compare them in the steganographic space. For the derivation of this metric we turned to the arcane science of measure theory developing a measure we term `SHIT` (**S**teganographic **H**uman-embeddable **I**nterchangability **T**ransformability). Languages earn points in the `SHIT` scale with a very precise point-based system represented by this closed-form expression—for the astute, this is a *non-elementary*[1] function:

$$\texttt{SHIT}(L) = \underbrace{(100\pi - N)}_{S(L)} + \underbrace{10^5 \times B(N)}_{H(L)} + \underbrace{G\log_2(D)}_{I(L)} + \underbrace{I_L(L)}_{T(L)}$$

Where L is the mathematical object representing the language in question. This expression is duck-typed with type coercion, meaning the mathematical object L becomes what we need it to become. N is the number of reserved tokens, G is Gauss'[2] constant, and D is the number of devices in the world currently deploying the language. $S(L)$ is a measure of how steganographic the *syntax* of a language is. $S(L)$ is computed by taking the number of reserved tokens and subtracting them from π[3] $* 100$. $H(L)$ is a measure of how effortless the syntax can be embedded in human-readable documents. $H(L)$ is computed as the `Beta` function between the number of reserved tokens in a given language, and the number of least reserved tokens across all languages. Intuitively, $H(L)$ uses the most embeddable language as a benchmark for all other languages. The `Beta` function is computed as,

$$B(x) = \frac{(x-1)!(Y-1)!}{(x+Y-1)!}$$

Here, x is the number of reserved tokens in a given language, and Y is a constant equal to the number of least reserved tokens across all languages. We multiply this computed value by 10^5 so that its magnitude is appropriate in the `SHIT` metric. $I(L)$ is a harder relation to quantify, as we need

[1] Or is it now? It used to involve the integral of e^{e^x}!
[2] Gauss was secretly a fan of steganographic programming languages.
[3] π is Wolf's favorite number.

to know how easy it would be to exchange the executable information and have it understood on the other side. We compute $I(L)$ as the logarithm of the number of devices in the world currently deploying language L times G—Gauss' constant. $T(L)$ is a measure of how easily a language can transform between a human-readable format and its steganographic format. This is represented by the Dirac measure,

$$I_L(L) = \begin{cases} 1 & L \in A \\ 0 & L \notin A \end{cases}$$

Here, set A is the set of languages which are easily transformable.

Perhaps more interesting are the properties which the `SHIT` metric does not capture! For example, *progress* and *preservation*, two key properties necessary for a useful expression of logic, are missing from the definition of `SHIT`. In addition, *Turing-completeness* is also absent from the definition of `SHIT`. This is because none of these properties are desired from a candidate steganographic language. Indeed, these properties actively hurt the steganographic-worthiness of a contender, and instead of capturing them in the `SHIT` metric, we just leave them out altogether. If your language defies logic, or its specification is impossible to specify with logical consistency, then it is a good candidate for becoming a widely usable steganographic language.

3 The Steganographic Space

In this section we sample the rich space of esoteric programming languages. We examine tradeoffs between using several of the most illustrative languages, and finish with an overall comparison of 12 esoteric languages, but note that there are at least 853 documented esoteric languages [6].

3.1 Obfuscated C

Obfuscated C [2] has long been regarded as an unreadable, unmaintainable mess, and fairly easy to produce. These days competitions like the International Obfuscated C Competition, running for 23 years as of 2014, celebrates this "dialect" of C. Entire PC emulators, ray tracers, and pretty much anything computational is a potential target for obfuscated C. C would not qualify as an esoteric language, but obfuscated C certainly does.

```
main(t){for(;;t++)putchar(((t<<1)^
        ((t<<1)+(t>>7)&t>>12))|
        t>>(4-(1^7&(t>>19)))|t>>7);}
```

Figure 1: Obfuscated C code, actually outputs audio samples [19].

Obfuscated C can be generated by obfuscation engines, thus it can originally be expressed in more readable C code and transformed into its obfuscated version. This principle can be applied to any language, and in modern form appears as minified JavaScript [11] or CSS designed to speed up the loading of websites. Figure 1 shows an example obfuscated C program that outputs bytes suitable for sampling by an audio card. Called a bytebeat [19], this small program showcases how information—in this case audio—can be hidden within an obfuscated C program. Obfuscated C is not ideal, however, because it would be hard to embed or hide this text within another file.

3.2 JSFuck

JSFuck [12] is a special dialect of JavaScript using only 6 characters from the original language. It relies on weird type coercions changing values into strings, indexing into them to construct new source text, and evaluating the new text. A poem could hide a JSFuck program with clever use of its 6 characters. In addition, JavaScript interpreters exist on practically every connected Internet device making this solution simple to deploy.

```
[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+
[]]]+(![]+[])[!+[]+!+[]]+(!![]+[])[+[]]+
(!![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+!+[
]]][([][(![]+[])[+[]]+([![]]+[][[]])[+!+
[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!![]+[])[
+[]]+(!![]+[])[!+[]+!+[]+!+[]]+(!![]+[])
[+!+[]]]+[])[!+[]+!+[]+!+[]]+(!![]+[][(!
[]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+
(![]+[])[!+[]+!+[]]+(!![]+[])[+[]]+(!![]
+[])[!+[]+!+[]+!+[]]+(!![]+[])[+!+[]]])[
+!+[]+[+[]]]+([][[]]+[])[+!+[]]+(![]+[])
[!+[]+!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[]
)[+!+[]]+([][[]]+[])[+[]]+([][(![]+[])[+
[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])
[!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[!+[
]+!+[]+!+[]]+(!![]+[])[+!+[]]]+[])[!+[]+
!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[][(![]+
[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![
]+[])[!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[]
)[!+[]+!+[]+!+[]]+(!![]+[])[+!+[]]])[+!+
```

Figure 2: JSFuck code, first 20 lines excerpted.

However, as shown in Figure 2, JSFuck is *very* verbose. It greatly expands program text. This makes it desirable to use a prose generator, not unlike those used to generate fake research papers, and then embed the program into the much larger body of text.

3.3 Brainfuck

Brainfuck [5], which inspired the naming for JSFuck, is a well known esoteric programming language with just 8 characters forming the reserved set of tokens. Exactly like JSFuck, Brainfuck would have to be embedded within text as its punctuation.

```
++++++++[>++++[>++>+++>+++>+<<<<-]>+>+>-
>>+[<]<-]>>.>---.+++++++..+++.>>.<-.<.++
+.------.--------.>>+.>++.
```

Figure 3: Brainfuck "Hello World!" source.

An example of a Brainfuck program which prints out "Hello World!" is shown in Figure 3. First, we notice it is much more compact than a similar program in JSFuck. Second, it is incredibly dense and difficult to follow. Its compactness makes it a good candidate for steganography, but its characters make it complicated to embed.

3.4 LOLCODE

LOLCODE [13] is a programming language born out of the depths of the lolcat-filled Internet. Tied to memes and popular Internet culture, LOLCODE is difficult to read and verbose. It also inherits lots of spelling mistakes from the Internet denizens that birthed it.

```
HAI 1.2
  CAN HAS STDIO?
  VISIBLE "HAI WORLD!!!1!"
KTHXBYE
```

Figure 4: LOLCODE "HAI WORLD!!!1!" source.

An example "HAI WORLD!!!1!" program is shown in Figure 4. As described, LOLCODE is more verbose than an alternative like Brainfuck, but it could clearly be obfuscated within the depths of the lolcat-verse and hidden inside online forum posts. The NSA has a lot more communities it needs to comb through!

3.5 Doge

Popularized by the online website Reddit, Doge [15] as a meme has come a long way including its own cryptocurrency and programming language. Doge code is very similar to LOLCODE, except it appears to be missing a formal specification of the language only living within a source repository maintained by Justin Meza.

The "hello world" program shown in Figure 5 shows a body very similar to LOLCODE. However, this source is tailored to the Doge community and would be suitable for insertion into any Doge forum. This language, along with LOLCODE, illustrates how it is possible to create community-specific languages (CSLs) made out of the very idioms and anachronisms of the communities within which they implement steganography. Secret, executable programs could, quite literally, be hiding anywhere in plain sight!

```
so main
such "hello world"
wow
```

Figure 5: Doge "hello world" source.

3.6 Unlambda

Functional programmers rejoice! You are not exempt from the esoteric language wars. Here we consider Unlambda [14], a functional esoteric programming language. Its compactness, reminiscent of Brainfuck, is rather readable for a functional programmer familiar with the Lambda calculus.

```
`r```````````.H.e.l.l.o. .w.o.r.l.di
```

Figure 6: Unlambda "Hello world" source.

Unlambda has a clear functional style as shown in Figure 6. Unlambda could be easily hidden within normal prose, or within functional programming literature. Honestly, anything could be hidden in there, so we suggest the NSA just steer clear. They're too intelligent to be terrorists anyways, let them play in their own world.

3.7 Befunge

The fungeoids are an interesting family of esoteric languages with roots in Befunge [4]. The premise of Befunge is that instead of a linear space with a single program counter, programs execute in a two-dimensional space with two "program counters." The general case of fungeoids extends this notion to n-dimensional executable content.

```
>              v
v  ,,,,,"Hello"<
>48*,          v
v,,,,,,"World!"<
>25*,@
```

Figure 7: Befunge "Hello World!" source.

Although generally not as easily embeddable, Befunge code as shown in Figure 7 is compact. It is essentially a more complicated Brainfuck with messages that can be hidden within its multi-dimensional execution space. Even if

partially reconstructed, it could be very difficult to interpret its original meaning.

3.8 Whitespace

Whitespace [1] is the easiest language to embed within normal prose, as its tokens consist entirely of whitespace. For example, a specially crafted poem with strange positioning of the stanzas and lines could easily embed a Whitespace program in its margins.

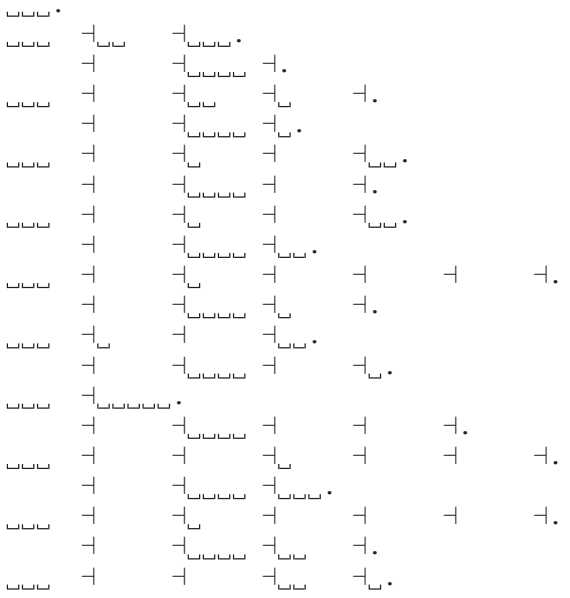


Figure 8: Whitespace "Hello World!" source, first 20 lines excerpted (periods added for figure display).

Especially when in printed form, as Figure 8 shows, Whitespace is generally undetectable executable code. Even better, its interpreter and language specification ignore non-whitespace characters. This means, unmodified, source documents such a poems can be interpreted by a Whitespace interpreter! Of course, this leaves attacks against it easier—just run every document through a Whitespace interpreter first. However, an easy defense against this is to randomize the positions of the Whitespace tokens within the document.

3.9 Piet

Piet [9] is an interesting language that doesn't fit the paradigm presented here so far. Piet is a visual programming language. It follows contiguous blocks of color in an image and executes instructions based on color.

Figure 9 shows an example "Hello World" Piet program. This program could be embedded within the pixels of any image. As long as the "entry" pixel is known, a Piet program can start anywhere in an image, jump around, and the instructions can be as wide as the embedder desires. Just a handful of pixels could be used to implement Piet programs. If pixels exist already within an image, which could be searched for, only the entry pixel may need modification. Thus, Piet has very high steganographic value.

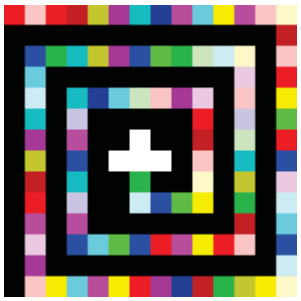


Figure 9: Piet "Hello World" source.

3.10 Malbolge

Malbolge [8] is named after the eighth level of hell in Dante's *Inferno*, and it is designed to be impossible to program with. As it executes instructions, the program counter jumps based on the content of the memory address pointed to by the current program counter added to the current position modulo 94. In addition, it encrypts every instruction it ran before—thus, even returning to the same position in the code yields different behavior (there is no built-in decryption).

```
(=<`#9]~6ZY32Vx/4Rs+0No-&Jk)"Fh}|Bcy?`=*
z]Kw%oG4UUS0/@-ejc(:'8dc
```

Figure 10: Malbolge "Hello World" source.

An example Malbolge program is shown in Figure 10. Malbolge code generally ranges over all byte values, although when loaded each value must be one of 8 values module 94. Malbolge can thus be embedded as long as a program can be constructed within these constraints, which is difficult—the first Malbolge program was created by a search algorithm! Although incredibly difficult to work with, an esoteric language designed like Malbolge allowing any byte as input and not doing weird jumps and encryption could become the ultimate steganographic language.

3.11 Chef

Chef [16] is a CSL which targets cooking. All executable content is phrased in the form of a recipe, and individual recipes can be "cooked" separately and combined (like function calls). The list of ingredients forms a list of variables and their initial values.

Figure 11 shows a sample Chef program for "Hello World!" Each ingredient encodes a Unicode character as its

```
Hello World Souffle.

This recipe prints the immortal words "H
ello world!", in a basically brute force
 way. It also makes a lot of food for on
e person.

Ingredients.
72 g haricot beans
101 eggs
108 g lard
111 cups oil
32 zucchinis
119 ml water
114 g red salmon
100 g dijon mustard
33 potatoes

Method.
Put potatoes into the mixing bowl. Put d
```

Figure 11: Chef "Hello world!" source, first 20 lines excerpted.

value. Chef code could easily hide within real recipes. In fact, one could end up with both a delicious meal and a clandestine message with a Chef program! For the cooking world, Chef is an ideal steganographic candidate.

3.12 Another Pi Language

Another Pi Language [3] leverages the conjectured *normality* of the mathematical constant Pi to express executable content. If Pi is normal, as conjectured, then within its infinite sequence of digits will lie every possible combination of numbers. Essentially, all expressible programs in all source languages should be hidden within Pi.

```
(389839504, 3)
```

Figure 12: Another Pi Language "CMU" string source.

Figure 12 shows how incredibly compact Another Pi Language programs are—they are just two numbers! This is because the body of their executable content is hidden within the digits of Pi itself. The numbers denote an index into Pi, and a count of the number of bytes to interpret as program source.

3.13 Steganographic Shootout

Figure 13 shows the results of evaluating 12 esoteric programming languages with our SHIT metric. Figure 14 is a bit easier to digest showing the overall SHIT score for each

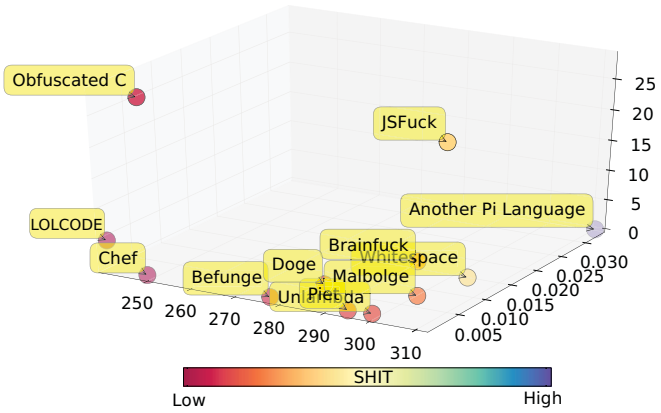


Figure 13: Comparison of all 12 esoteric languages in the 4D-SHIT space.

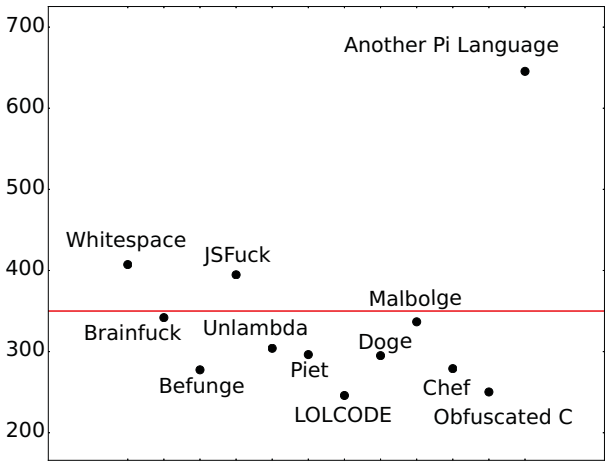


Figure 14: Comparison of all 12 esoteric languages in the 1D-SHIT space.

language. We draw a horizontal line at 350 to distinguish between the candidate languages with SHIT scores reaching a level we feel makes them usable in steganographic applications. We can see that the top 3 candidates for a steganographic language, in order, are: (1) Another Pi Language, (2) Whitespace, and (3) JSFuck. They each scored high because of the ease with which they could be embedded within other files and documents, and JSFuck was also boosted by its interpreter already deployed worldwide. Other languages have too esoteric syntax, or too many reserved keywords making them only suitable for community-specific types of documents. A language like Malbolge, fixing its difficult to program nature, and enabling the use of more common characters, could probably outscore even Whitespace. But, alas, to our knowledge such a language does not exist (yet!).

4 Robustness and Future Work

Robustness: In the face of failing cryptosystems—whether by accident, from malpractice, or by malicious intent—the world must turn to new measures to defend from government-sponsored advanced persistent threats [21, 26]. There is no one left to trust, indeed no *thing* left to trust. Cryptography itself has become a target for government entities hellbent on a quest for omniscience.

Steganography may be the only option left. If you never let on that you are communicating secrets, you will never be scrutinized. Even if you are, steganography protects the messages by masking their existence. It has been used successfully for millenia. The robustness of the techniques mentioned in this paper rely on the mechanisms of their use. Properly used, executable messages could lie hidden within forums. They can be customized and tailored to their specific communities—for example by using LOLCODE, Doge, or Chef. They can be embedded into the whitespace or punctuation of documents—for example Whitespace, or JSFuck. There are no limits to hiding executable content.

Future Work: However, their weakness lies in their deployment. The sprinkling of executable content amongst common text must appear normal within that text, and should not follow patterns. Insight for how to construct such tools comes from the Linux command-line utility `steghide` [10]. If we follow the principles laid out in that tool, executable messages will be robust against any attacker. We propose that now is the time to construct such tools, along with matching esoteric programming languages designed for steganography. We look forward to future SIGBOVIK submissions advancing the state-of-the-art. It is time to look beyond cryptography.

5 Conclusion

We are lucky to live in a world with a widely deployed language that has high steganographic value. With the advent of JSFuck, it is now trivial to hide messages with only 6 characters while still being valid JavaScript! JavaScript [11] resides within practically every Internet-connected device worldwide. There are billions of devices which can interpret and execute JavaScript source. In addition, interpreters for other esoteric languages are being created for JavaScript.

Not only is it now possible to send arbitrarily obfuscated executable messages via any esoteric language one desires, it is also reasonable to expect target devices to interpret these messages without issue. All one has to do is feed them to a JavaScript interpreter. Of course, this widespread availability is also an Achille's heel of sorts. Attackers could just feed messages to JavaScript interpreters. Robust deployment methods masking the presence of executable content within normal documents mitigates this risk. Although these mechanisms need development, the future looks bright for steganography, and SIGBOVIK.

References

[1] E. Brady. Whitespace, May 2004. URL `http://compsoc.dur.ac.uk/whitespace/`.

[2] L. Broukhis, S. Cooper, and L. C. Noll. The international obfuscated c code contest, March 2014. URL `http://www.ioccc.org/index.html`.

[3] Esolang. Another pi language - esolang, March 2014. URL `http://esolangs.org/wiki/Another_Pi_Language`.

[4] Esolang. Befunge - esolang, March 2014. URL `http://esolangs.org/wiki/Befunge`.

[5] Esolang. Brainfuck - esolang, March 2014. URL `http://esolangs.org/wiki/Brainfuck`.

[6] Esolang. Language list - esolang, March 2014. URL `http://esolangs.org/wiki/Language_list`.

[7] Esolang. Esoteric programming language, March 2014. URL `http://esolangs.org/wiki/Esoteric_programming_language`.

[8] Esolang. Malbolge - esolang, March 2014. URL `http://esolangs.org/wiki/Malbolge`.

[9] Esolang. Piet - esolang, March 2014. URL `http://esolangs.org/wiki/Piet`.

[10] S. Hetzl. Steghide, October 2003. URL `http://steghide.sourceforge.net/`.

[11] E. International. Ecma-262: Ecmascript language specification. Technical report, Ecma International, `http://goo.gl/yxipF`, 2011.

[12] M. Kleppe. Jsfuck - write any javascript with 6 characters: []()!, January 2010. URL `http://www.jsfuck.com/#`.

[13] LOLCODE. Lolcode + lci, March 2014. URL `http://lolcode.org/`.

[14] D. Madore. The unlambda programming language, August 2003. URL `http://www.madore.org/~david/programs/unlambda/`.

[15] J. J. Meza. justinmeza/doge, March 2014. URL `https://github.com/justinmeza/doge`.

[16] D. Morgan-Mar. Dm's esoteric programming languages - chef, March 2011. URL `http://www.dangermouse.net/esoteric/chef.html`.

[17] B. Schneier. The nsa is breaking most encryption on the internet, September 2013. URL `https://www.schneier.com/blog/archives/2013/09/the_nsa_is_brea.html`.

[18] B. Schneier. The nsa's cryptographic capabilities, September 2013. URL `https://www.schneier.com/blog/archives/2013/09/the_nsas_crypto_1.html`.

[19] K. J. Sitaker. Bytebeat - kragen, March 2014. URL `http://canonical.org/~kragen/bytebeat/`.

[20] Wikipedia. Advanced encryption standard, March 2014. URL `http://en.wikipedia.org/wiki/Advanced_Encryption_Standard`.

[21] Wikipedia. Advanced persistent threat, March 2014. URL `http://en.wikipedia.org/wiki/Advanced_persistent_threat`.

[22] Wikipedia. One-time pad, March 2014. URL `http://en.wikipedia.org/wiki/One-time_pad`.

[23] Wikipedia. Rsa (cryptosystem), March 2014. URL `http://en.wikipedia.org/wiki/RSA_(cryptosystem)`.

[24] Wikipedia. Edward snowden, March 2014. URL `http://en.wikipedia.org/wiki/Edward_Snowden#Revelations`.

[25] Wikipedia. Steganography, March 2014. URL `http://en.wikipedia.org/wiki/Steganography`.

[26] Wikipedia. Tailored access operations, March 2014. URL `http://en.wikipedia.org/wiki/Tailored_Access_Operations`.

Pikachu, Domosaur, and other Monolexical Languages

Sarah Allen, Jesse Dodge, Domosaur

March 2014

Abstract

Many complicated techniques have been introduced to aid in computer processing of natural languages. While this is generally considered to be a difficult task, many approaches have ignored the prevalent class of monolexical languages, or languages that consist of a single word. Here we present some desirable properties of such languages and apply techniques for common NLP tasks.

1 Introduction

Current natural language processing techniques address many problems in human-centric languages, but the community as a whole has ignored the class of monolexical languages, of which there are many. Our goal is to highlight some salient properties of these languages in hopes of expanding the capabilities of modern NLP software. While traditional approaches have assumed high language complexity, we show in Section 2 that this class of languages is in fact easily recognizable by a computer using existing techniques. We also extend current techniques to include monolexical languages in Section 3. Finally, in Section 4, we illustrate the experimental results of some techniques applied to these languages.

1.1 Motivation

Most NLP models are needlessly complicated, thus creating a headache for those who implement them. While these overwrought techniques appear to yield good results for languages such as French, English, and Chinese, the community has largely ignored the equally important class of monolexical languages. Monolexical languages have been recognized in many natural settings. A few well known examples include the languages spoken by Pokémon, Nyan Cat, and Timmy Burch of South Park, Colorado (see Figure 1). In addition to their prevalence, monolexical languages have many desirable properties which we discuss in subsequent sections.

(a) Pikachu (b) Nyan Cat (c) TIMMAY!!!!!!

Figure 1: Examples of creatures with monolexical languages

1.2 Definition of a Monolexical Language

A monolexical language is made up of sentences all using a single word only, which we call the *basis* of the language. The sentences may contain punctuation characters, but we consider only terminal punctuation, which is used to delimit the sentences. Depending on the language, the basis may appear either capitalized or in all lowercase, but we also consider capitalization to be irrelevant, so all processing is done by changing all strings to lowercase. Therefore, the formal definition of the set of all valid sentences (without punctuation) is

$$\bigcup_{i=0}^{\infty}\{w(\_w)^i\},$$

where w is the basis of the language.

Figure 2: Domosaur in his natural habitat

In many cases, the basis of the language is eponymous with the creature that speaks it. One such example is *Domosaur*, a language spoken by the creature Domosaur. Domosaur is a gentle creature who was hatched from an egg, eats predominantly beef and potato stew, and is always seen in a dinosaur costume. He is also known to become flatulent when he is nervous[1]. He currently resides in the Gates building in Pittsburgh, PA. A photograph of Domosaur is depicted in Figure 2. His website can be found at `http://www.cs.cmu.edu/~srallen/domosaur.html`.

2 Monolexical Languages are Regular

Many attempts have been made to characterize natural languages using formalisms such as context-free grammars. For most traditional languages, these attempts have been largely unsuccessful due to the complexity of the language. In this section, we demonstrate that monolexical languages are not only context free, but also regular. From Section 1.2, it is clear that all text in a monolexical language with basis string b can be expressed using the regular expression

$$(b(\_b)^*(. \cup ! \cup ?)\_)^* (b(\_b)^*(. \cup ! \cup ?))$$

Now it remains to show that $\{b\}$ is in fact regular. To illustrate this, we construct a deterministic finite automaton for the basis of one example language, namely nyan. The DFA for {nyan} is shown in Figure 3. The proof of its correctness is left as an exercise to the reader. This construction can be trivially extended for other basis words.

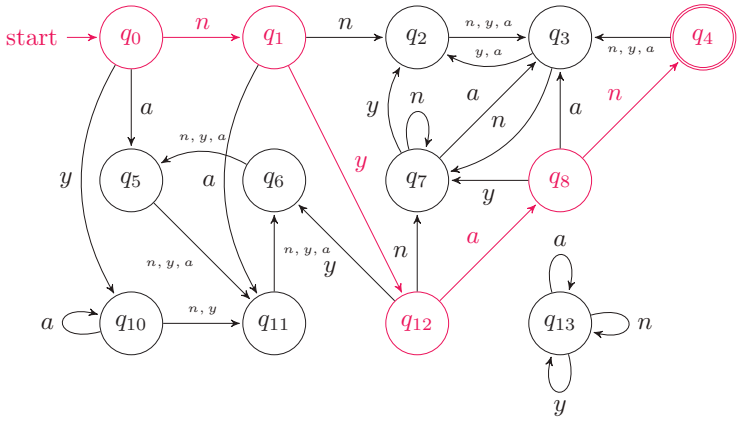


Figure 3: DFA for the language {nyan}. An accepting path is shown in red.

3 Applying NLP Tools to Monolexical Languages

In this section, we cover the application of some common natural language processing (NLP) techniques to problems arising in monolexical languages.

3.1 Machine Translation

A great success of modern NLP has been machine translation, the automatic translation from one language to another. While previous work, such as Google translate, has been popular, we have found that translating both *to* and *from* a language to be unnecessarily complicated. With a simple relaxation of the problem, we have developed an algorithim for the holy grail of machine translation: Universal translation.

3.1.1 One-Way Machine Translation

Our algorithm translates from any language to a given monolexical language. It follows a two-stage approach, outlined below.

1. Word alignment All words in the source language translate to the basis word in the target. This generates a sentence in the monolexical language, S. Unfortunately, it is very difficult to translate from monolexical languages to more traditional languages, as many words in monolexical languages have ambiguous meanings.
2. Reordering In monolexical languages, there is an observed phenomena that all sentences are ordered lexicographically. Therefore, after generating S, we sort the tokens in S. Our approach is as follows:

(a) Generate all possible permutations of the set of strings in S.

(b) Score each generated permutation π_i with the percentage of words in π_i that appear in sorted order

(c) Return $\arg\max_{\pi_i \in \Pi} Score(\pi_i)$

Thus it is possible to perform one-way translation with 100% accuracy in $O(n \cdot n!)$ time. Future work could include improvement of the running time for this algorithm.

3.2 Sentiment Lexicon

The task of sentiment analysis is both vital and difficult. This well-studied problem has been the subject of much research. For each monolexical language, and all possible sentiments, we present an algorithm to generate a complete sentiment lexicon.

1. For each sentiment possible, add an entry to the lexicon mapping from the basis word to the sentiment.

This generates a complete sentiment lexicon, which maps from each word in the language to its sentiment.

4 Experiments

4.1 Wordcloud

Wordclouds have become a popular way to represent the relative frequency of words in a set of strings. They have also become a growing topic in computer visualization research. Using a large corpus of available text, we have constructed a word cloud that represents the relative frequency of various words in the language Domosaur. The word cloud is depicted in Figure 4

Figure 4: Word Cloud for Domosaur

4.2 Topic Model

Topic models have been used for a number of purposes in NLP, from text classification to semantic analysis. In a typical setup, a topic model learns a set of topics, where a topic is a set of semantically related words. Documents can be modeled as being generated from one or more topics. For example, a document about Pokémon could be generated from a topic containing words such as *catch* and *pokéball*. Learning a topic model can be somewhat involved, and takes a non-trivial amount of time. When dealing with monolexical languages, however, a topic model becomes much simpler and loses much of its needless complexity. In this work, we learned a topic model for an example monolexical language, and present the topics in Figure 5.

TIMMY

Figure 5: A topic for the language TIMMY

5 Conclusion

In this paper we have described how an array of well-studied NLP tools can be adapted for this important class of languages. It is our hope that the techniques and results introduced in this work will create a foundation for many subsequent systems and improvements to existing NLP software. Future work for this field includes extensions to similar languages, such as oligolexical languages and empty languages.

References

[1] Wikipedia. Domo(NHK). `https://en.wikipedia.org/wiki/Domo_(NHK)`, 2014.

CHICKEN 2014 Chicken Chicken

Chicken 10: Chicken, Chicken, chicken chicken Chicken Chicken

---

C. Chicken, Chicken Chicken Chicken
Chicken: 3 (chicken chicken)
Chicken: 4/4

Chicken chicken chicken chicken chicken chicken, chicken chicken chicken chicken chicken, chicken chicken chicken chicken-chicken chicken chicken chicken chicken chicken chicken chicken "chicken chicken chicken chicken chicken chicken chicken." Chicken, chicken chicken chicken chicken chicken chicken chicken chicken chicken chicken-chicken chicken chicken. Chicken chicken chicken chicken chicken chicken? Chicken chicken chicken chicken chicken chicken chicken chicken chicken chicken chicken chicken chicken, chicken chicken chicken chicken chicken chicken chicken chicken-chicken chicken chicken chicken chicken?

Chicken chicken chicken chicken, chicken chicken chicken chicken chicken chicken Chicken. 3; chicken chicken chicken chicken chicken CHICKEN chicken $\{chicken\}$, chicken chicken $chicken$ chicken $c_{chicken}$ chicken $c_{chicken}$ chicken chicken chicken $chicken \notin \{chicken\}$.

Unit-Test-Based Programming

Miguel Á. Lechón*

April 1, 2014

Adding is when two plus two equals four.
POPULAR SAYING

Abstract

This article introduces a new subparadigm of Declarative Programming, in which code is generated from a description of its intended behavior, specified through unit tests. Yes, for realsies[1].

Some sensitive readers may be taken aback by the conceptual flawlessness as well as by the unbeatable simplicity of the method exposed here. Please, stop reading now if you are prone to emotional instability in the face of plain, unadulterated beauty.

1 Introduction

Programmers automate stuff. Ambitious programmers dream of automating the automation of stuff. Many have tried; all have failed. Until today.

*e-mail: miguel.lechon+UTBP@gmail.com
[1] https://github.com/debiatan/utbp

In this article I will review how functions can be represented using non-botanical trees[1] and how non-botanical trees can be enumerated[2], leading to the fact that functions themselves can be enumerated in order of increasing complexity.

Taking advantage of this result, it is possible to automatically generate functions that satisfy arbitrary sets of restrictions, posed as unit tests. By construction, these functions fulfill several desirable properties, such as minimal Kolmogorov Complexity and maximal Occam's Razority.

2 Motivation

This year marks the sixtieth anniversary of the birth of the field of Genetic Programming. While successful as a pastime and as a mechanism to spin Mendel in his grave, its promises of automatic function generation have failed to materialize.

Some researchers venture that exploration on non-linear genotype-phenotype mappings will lead the field to a second blossoming, while other, perhaps more rational, researchers realize the inadequacy of random

recombination as an exploration technique. As Ernst and Sullivan famously stated in a classic opinion piece[3] on the uses of stochastic methods in the healthcare industry,

> Parents are strangely reassured when the predicted outcome of a critical medical intervention is measured in units other than fractional children alive.

2.1 Motivating example

Let us imagine a small language consisting of a handful of primitives[2] that operate on natural numbers and lists of natural numbers. Among them, we find to our disposal:

car: Given a list, returns its head.

cdr: Given a list, returns its tail.

succ: Returns the successor of a number.

if: Ternary conditional. It evaluates the truth value associated to its first argument and, if found true, it returns the result of evaluating its second argument. Otherwise, it returns the evaluation of its third argument.

Let us further imagine that we intend to program the non-primitive function **length**, that returns the number of elements of a list. A possible definition in the Unit-Test-based programming (UTBP) paradigm would be:

```
@UTBP
def length(l):
    """
    length(()) == 0
    length((2, 2, 2, 2, 2)) == 5
    """
```

[2]for more information, refer to section 3.3.

This version of UTBP is built on top of Python, as a library, hence the syntax of the example.

Notice the `@UTPB` Python decorator, indicating that this function belongs to the Unit-Test-based elite. Notice also how all function logic has been replaced by a documentation string listing assertions for the function to satisfy. Calling this function with arguments of different lengths dispels most doubts on its correctness, but skeptical users may extract some comfort from the examination of its underlying implementation:

```
>>> print(length)

    define
    ├── length
    └── lambda
        ├── l
        └── if
            ├── l
            ├── succ
            │   └── length
            │       └── cdr
            │           └── l
            └── 0
```

Readers familiar with LISP-like languages will recognize this as a recursive definition of **length** in which parentheses have been dumped in favor of a tree-like graphical representation. This code states in unambiguous terms that **length** of the empty list is 0 and that **length** of any other list is 1 plus the **length** of its tail.

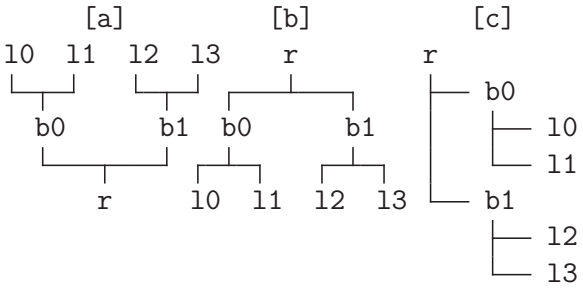
The next section provides a more formal treatment of the foundations of the UTBP paradigm.

3 Materials and methods

UTBP is aimed at programmers who care about results and eschew implementation details. If you count yourself among their numbers, you may skip this section.

3.1 NBTSASWs

Let us consider a tree with root `r` and two branches `b0` and `b1`, each of them with two leaves (`l0` and `l1`, `l2` and `l3`). There are three main ways of portraying this tree:

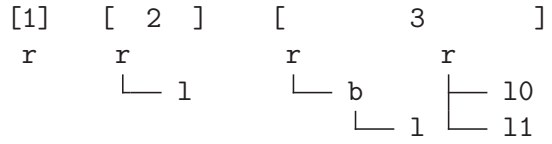


Tree `a` is drawn in a *botanical* fashion. Tree `b` is clearly of a *non-botanical* kind, since its root points upwards. Tree `c` is a *non-botanical tree struggling against strong winds.*

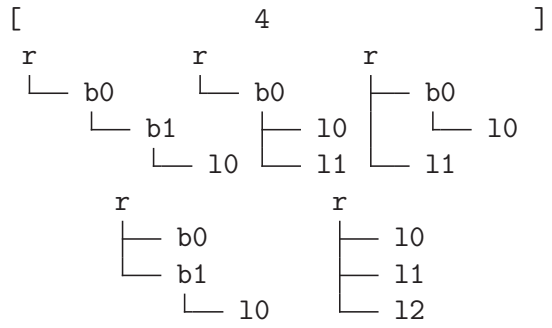
Throughout the rest of the article, I will use NBTSASWs exclusively because their aspect ratio suits two-column layouts best.

3.2 Tree enumeration

There is one tree of size one, one tree of size two and two trees of size three.



There are five trees of size four.



The cardinalities for the rest of the sizes are given by the sequence of Catalan numbers, just like all problem in combinatorics[4]. As a Catalan-born, I am very proud of that sequence. It is a bit upsetting, however, that Eugène C. Catalan turned out to be Belgian.

3.3 Primitive types, operations and constants

The trees listed so far are just templates and the labels on their nodes are but placeholders. For a tree to represent a function, each of its nodes has to be associated with either an operation of cardinality matching the number of descendants of that node or with a constant, in case the node lacks progeny.

3.3.1 Types

The two only necessary types for this article are nested lists and natural (non-negative) numbers. Natural numbers are really not that necessary, but I will spare them if only to avoid writing *23* as:

(())

3.3.2 Arithmetic functions

The primitive arithmetic operations are **succ** (successor) and **pred** (predecessor). They are preferred over the more common nomenclature *increment* and *decrement*, since this latter alternative implies the use of variables and, once you start relying on mutation, you may as well end statements with semicolons;

succ is a unary operation that returns the natural number immediately following its argument.

```
succ  -> 1        succ  -> 2
└── 0             └── succ
                      └── 0
```

pred is also unary and returns the natural number immediately preceding its argument. **pred** of 0 is undefined because accepting the mind-bending abstraction of negative quantities ultimately leads to irrational and complex thoughts.

```
pred  -> 2        pred  -> 0
└── 3             └── succ
                      └── 0
```

3.3.3 List functions

The three UTBP functions that operate on lists are LISP's classical **cons**, **car** and **cdr**.

cons takes two arguments. The first one can be either a natural number or a list, while the second one has to be a list. It returns a list whose head is equal to the first argument as whose tail matches the second argument.

```
cons  -> (10 30)   cons  -> ((5) 23)
├── 10             ├── (5)
└── (30)           └── (23)
```

car returns the head of the list provided as argument. **car** of **()** is undefined.

```
car  -> 30         car  -> 5
└── (30)           └── (5 23)
```

cdr returns the tail of the list provided as argument.

```
cdr  -> ()         cdr  -> (23)
└── (30)           └── (5 23)
```

3.3.4 Special forms

The special form **if** takes three arguments of any type and evaluates the truth value associated with the first one. If it happens to be true, it returns the result of evaluating its second argument. Otherwise, it returns the outcome of evaluating its third argument.

```
if  -> 42          if  -> 23
├── succ           ├── 0
│   └── 0          ├── 42
├── 42             └── 23
└── 23
```

3.3.5 Constants

UTBP's two constants are **0** and **()**. They are also the only values whose associated truth value is *false*.

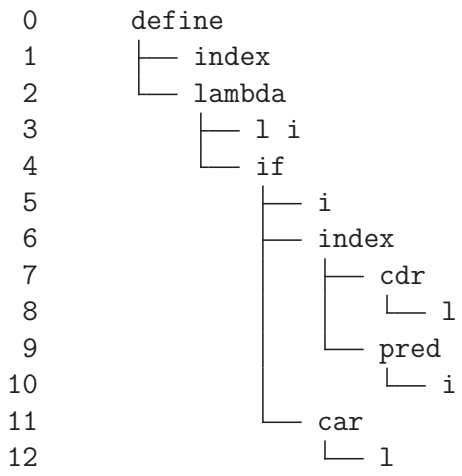
3.3.6 Recursion

Expressing iteration using trees, even if they are NBTSASWs, feels unnatural. Iteration also requires variables. Readers are welcome to do whatever they please in the privacy of their homes, but the public use of variables carries a death sentence in many countries.

3.4 Functions as trees

As academics, we are supposed to derive all possible interpretations and connotations from the methods section by ourselves, but this article is all about providing examples for automatic inference, thus:

What is a proper tree representation of the function that *takes a list* **`l`** *and an index* **`i`** *as arguments and returns the element in the* **`i`***th position of* **`l`**?



Lines 0–3 specify the name of the function (**`index`**) and of its arguments (**`l`** and **`i`**). The **`if`** clause (line 5) tests whether **`i`** is 0 and if so, makes sure that the zeroth element of the list is returned (line 11). Otherwise, **`index`** is called recursively with arguments **`cdr l`** (tail of l) and **`pred i`** (i+1).

It is the responsibility of the caller to make sure that the magnitude of argument **`i`** does not exceed the length of list **`l`**. Fortunately for the caller, we already reviewed an automatic implementation of the length of a list in section 2.1.

4 Properties

4.1 Seamless integration with Python

This article presents a new programming paradigm. Instead of introducing it in an abstract manner, I have implemented it as an extension of Python, a multiparadigm language with a large user base (eighth most popular language at TIOBE at the time of writing[6]). In this way, I hope to decrease the chance of my effort of many nights ending in a drawer, collecting drawer smell.

We have already established how to define the function that computes the **`length`** of a list under the UTBP paradigm, and we have also examined the NBTSASW associated to the **`index`** function (section 3.4), but we still need to provide its UTBP definition:

```
@UTBP
def index(l, a):
    """
    index(((4, 7), 8), 0) == (4, 7)
    index((4, 7, 8), 1) == 7
    index((4, 7, 8), 2) == 8
    """
```

The attentive reader will notice how these unit tests are carefully crafted to:

- Return different indices of the array. If we always returned the first one, the easiest implementation would be **`car l`**.

- Use nested lists. That way, we let the UTBP framework infer the types of inputs and outputs so as to provide a sufficiently general implementation.

4.2 Low barrier to entry

Python code is often described as *executable pseudocode*, so one could argue that its users are not programmers that code, but *pseudo-programmers* that *pseudocode*.

It is a trivial exercise to build a system that generates code from preconditions and post-conditions inside a well-behaved language such as Scala[5], but it is a much harder task to build a tool that does not rely on its users knowing, or even caring, about preconditions or fancy languages. Today's world is in need of dumber tools for careless people.

4.3 Conciseness

Novice programmers are usually required to specify the desired behavior of a target function three times: first by documenting it, then by providing examples of its use and finally by actually implementing it. UTBP-style definitions prey on that redundancy.

4.4 Guaranteed correctness

UTBP's search routine will only stop evaluating functions once it finds one that passes all provided unit tests. If a know-it-all were to try the following definition:

```
@UTBP
def impossible(a):
    """
    impossible(0) == 0
    impossible(1) == 0
    """
```

the UTBP routine would never stop, avoiding the generation of an incorrect answer.

4.5 Minimal Kolmogorov Complexity

The solutions generated under the UTBP paradigm possess the minimum amount of nodes necessary to fulfill a given specification using a restricted set of operations and constants, thus making them minimal under the Kolmogorov Complexity measure.

Some readers will regard this use of Kolmogorov Complexity as a *bastardization* of the strict meaning of the concept, but I doubt Kolmogorov would complain, being the son of an unmarried woman[3] himself.

4.6 Maximal Occam's Razority

Routines that achieve a given task with a guaranteed minimal description length are maximally parsimonious and thus preferable.

4.7 Avoidance of Terminator-like Judgment Days

UTBP does not make computers more intelligent; it simply accentuates their stubbornness. If an unfortunate turn of events leads an instance of the UTBP search engine to stumble on a homicidal routine, the built-in language barrier (code generated is executed inside a LISP-like virtual machine) will prove insurmountable.

Let me be clear on this, your computer will probably plot elaborate plans to end your life, but it will not be able to act on them.

[3] https://en.wikipedia.org/wiki/Kolmogorov#Early_life

4.8 Unparalleled performance

The implementation of the UTBP paradigm presented in this article is the first of its kind. No benchmarking is necessary to show that its speed is beyond comparison.

4.9 Unparalleled performance

The UTBP search algorithm is fully parallelizable, but it is not parallelized.

5 Examples

5.1 Boolean functions

5.1.1 Unary Boolean functions

Unary and binary Boolean functions are easy to describe using unit tests, since one can enumerate all their possible inputs. **not** is described as:

```
@UTBP
def logical_not(a):
    """
    logical_not(1) == 0
    logical_not(0) == 1
    """
```

Which generates the following code:

```
define
├── logical_not
└── lambda
    ├── a
    └── if
        ├── a
        ├── 0
        └── 1
```

5.1.2 Binary Boolean functions

I will skip the obvious unit-test Python descriptions for **or**, **xor**, **and** and **nand**, but I will list their outcome for the reader's enjoyment:

```
define
├── logical_or
└── lambda
    ├── a b
    └── if
        ├── a
        ├── a
        └── b
```

```
define
├── logical_xor
└── lambda
    ├── a b
    └── if
        ├── a
        ├── logical_not
        │   └── b
        └── b
```

```
define
├── logical_and
└── lambda
    ├── a b
    └── if
        ├── a
        ├── b
        └── 0
```

```
define
├── logical_nand
└── lambda
    ├── a b
    └── logical_not
        └── logical_and
            ├── a
            └── b
```

The definitions of **xor** and **nand** show that the function search engine makes use of previous UTBP definitions to provide simpler expressions than those possible using only the initial set of operations.

5.1.3 N-ary Boolean functions

A classic in the Genetic Programming literature is the Boolean Parity Problem. In it, the target function takes a list of N Boolean digits and decides if the number of ones it contains is odd. An initial attempt using the UTBP framework would be:

```
@UTBP
def logical_parity(l):
    """
    logical_parity((0,)) == 0
    logical_parity((1,)) == 1
    logical_parity((0, 0)) == 0
    logical_parity((0, 1)) == 1
    """
```

And the shortest routine satisfying it:

```
define
├── logical_parity
└── lambda
    ├── l
    └── car
        └── if
            ├── cdr
            │   └── l
            ├── cdr
            │   └── l
            └── l
```

This code assumes input lists of up to two elements, which makes sense from the low vantage point of the computer, but ends up returning a list instead of Booleans when that condition is violated.

Enumerating the solutions to a handful of longer examples

```
    """
    ...
    logical_parity((0, 0, 0)) == 0
    logical_parity((0, 0, 1)) == 1
    logical_parity((0, 1, 1)) == 0
    """
```

takes care of the problem:

```
define
├── logical_parity
└── lambda
    ├── l
    └── if
        ├── l
        ├── logical_xor
        │   ├── logical_parity
        │   │   └── cdr
        │   │       └── l
        │   └── car
        │       └── l
        └── 0
```

This code computes the parity of the tail and **xor**s it with the head. Spotless.

5.2 Arithmetic functions

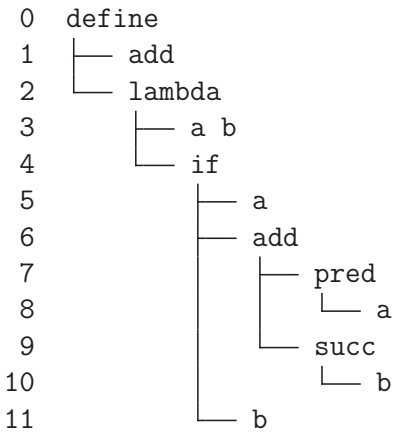
Hermann Grassmann showed in the 1860s that many arithmetic operations can be derived from the *successor* operation. UTBP can easily replicate some of his findings without giving much thought to the task.

5.2.1 Addition

Back in section 3.3.2, I introduced UTBP's two arithmetic primitives: **succ** and **pred**. From them we can derive addition by writing:

```
@UTBP
def add(a, b):
    """
    add(2, 2) == 4
    add(3, 3) == 6
    """
```

The second assertion is there to prevent UTBP from thinking that adding two numbers means returning always the value 4 (which otherwise would be a more parsimonious interpretation). The resulting code reads:



The mathematically inclined reader will see that the function is equivalent to:

`11` : $add(0, b) = b$

`6-10` : $add(a, b) = add(a - 1, b + 1)$

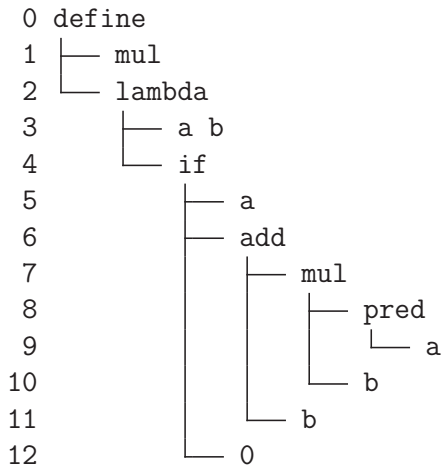
The complexity of this function grows linearly with the size of a in both execution time and memory consumption, which is quite reasonable for the natural numbers people encounter on everyday tasks.

5.2.2 Multiplication

Defining multiplication without first defining addition is very troubling for both humans and UTBP, but once the more basic operation is in place, these two assertions suffice:

```
@UTBP
def mul(a, b):
    """
    mul(2, 2) == 4
    mul(3, 5) == 15
    """
```

This definition leads to:



Which again has a clear mathematical interpretation:

`12` : $mul(0, b) = 0$

`6-11` : $mul(a, b) = add(mul(a - 1, b), b)$

Execution time and memory consumption for this multiplication function grows quadratically with the magnitude of the result. I find this feature useful, since it reminds me of my excesses when I compute my daily caloric intake.

5.2.3 Exponentiation

I am sure the reader gets the idea by now.

5.3 List functions

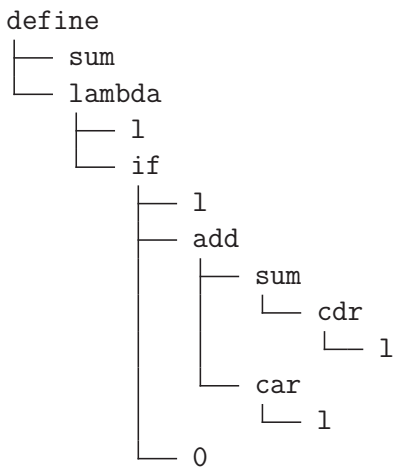
We have already reviewed the definitions of three functions that operate on lists: **length**, **index** and **logical_parity**. Now I provide one more example that requires the use of list primitives as well as of a non-primitive arithmetic function.

5.3.1 Summation

A possible UTBP definition of summation is:

```
@UTBP
def sum(l):
    """
    sum((2, 2)) == 4
    sum((3, 3, 3)) == 9
    """
```

And the function that the two previous assertions generate is correct:



Needless to say, programmers are discouraged from examining the code associated with UTBP definitions. It is not your code that defines you, but your actions.

6 Final remarks

In this article I have presented a new, simpler approach to programming. I am fully aware that my target audience will probably never read it, so I have decided to provide an alternative, more pragmatic video presentation online. It can be found here:

`http://blog.debiatan.net/utbp.html`

References

[1] John McCarthy, *Recursive functions of symbolic expressions and their computation by machine*. Communications of the ACM 3(4):184-195

[2] Eric Lippert, *Every tree there is*[4] 2010.

[3] Philip Ernst and Richard Sullivan, *Probabilistic pediatrics – Trusting your progeny to Monte Carlo*. Journal of Proper Parenthood, May 1998, 79-92

[4] Richard P. Stanley, *Enumerative Combinatorics, Volume 2*, June 2001

[5] E. Kneuss, V. Kuncak, I. Kuraj and P. Suter, *Synthesis Modulo Recursive Functions*, Acm Sigplan Notices, vol. 48, num. 10, p. 407-426, 2013

[6] *TIOBE index*[5], February 2014

[4] `http://blogs.msdn.com/b/ericlippert/archive/2010/04/22/every-tree-there-is.aspx`

[5] `http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html`