SIGBOVIK 2007
Proceedings

April 1^{st} 2007
Carnegie Mellon University
Pittsburgh, Pennsylvania USA
http://sigbovik.org/

A Message From the Program Committee

The Association for Computational Heresy Special Special Interest Group (ACH SIGBOVIK) on Harry Q. Bovik's 6th Binarennial Workshop about Symposium on Robot Dance Party of Conference in Celebration of Harry Bovik's birthday has a rich and noble legacy. We, the current members of the program committee were delighted to discover this legacy as we ~~made it up~~ rooted through the files of the SIGBOVIK Ministry of History, writing this introduction.

The first SIGBOVIK Conference was held on young Harry Bovik's first birthday in 1942. It was a brief affair, as all of the attendees had short, childish attentions spans. This has not changed. This was the occasion of Fred Hacker's famous talk, *Waaaah: Want Bottle*. This talk set a precedent for SIGBOVIK's level of discourse through the years.

The SIGBOVIK Conference was not held on Harry's second and fourth birthdays. These are known colloquially as the dark years. Harry was going through a *phase* at the time. This phase went on for six years. Even then, Harry had a long-term approach that would serve him well in his research.

SIGBOVIK IV: BOVIKMANIA was a wild affair. Chuckles the Clown's keynote was interrupted by what records describe only as *The Cake Incident.* Records are unclear about the nature of this incident, but we think it is safe to say it involved cake. This conference also marked Harry Bovik and Fred Hacker's first public feud, signified by a pair of position papers titled *Fred Hacker is a doodyhead* and *Takes one to know one.*

The fifth SIGBOVIK symposium helped establish the conference as the foremost venue for computer science research and Harry Bovik himself as its brightest star, noted for his clever ideas and stunning seventies wardrobe. Of the eighteen presenters, seventeen went on to win Turing awards. The other was current Governor of California Arnold Schwarzenegger. However, the spotlight was stolen by Harry himself who submitted an astonishing thirty-two papers to the conference in honor of his age. Each was a veritable oyster's pearl of brilliance. Unfortunately, they were lost in an accident involving three radishes, four peeled carrots, six celery sticks, and a cuisinart machine. Demoralized, Harry never replicated those results. Computer Science still has not recovered from this loss.

This brings us to up to date, ready for the sixth SIGBOVIK Conference and Harry's sixty-fourth birthday. The minds behind this year's crop of SIGBOVIK submissions are so bright, they wear sunglasses to bed. We, on behalf of the SIGBOVIK Organizing Committee, are proud to present this year's submissions.

-The SIGBOVIK Committee For Writing An Introduction to the Proceedings

Contents

Track I:

Psychopathology and Logic

A non-judgmental reconstruction of drunken logic

Robert J. Simmons*

Keywords: Lax logic, ex-lax logic, handwaving logic, drunken logic, shotglass monad, durnken logic, chemically assisted reasoning, alcohol in computer science

We investigate the extension of previous work by Krishnaswami et al. in [6] on Handwaving Logic, a logic that can be effectively modeled by Fairtlough and Mendler's Lax Logic [4], towards trying to achieve a reasonable formalization of "drunken logic." More advanced formalizations of drunken logic fail to be modeled effectively by lax logic, and we argue that much more study deserves to be paid to this and other concerns which we group together under the umbrella of Chemically Assisted Reasoning (CAR - but don't drink and drive). However, unlike various "judgmental" reconstructions, for instance of modal and lax logic [8], this will *not* be a judgmental reconstruction. We're not here to judge, man.

Section 1 briefly re-presents handwaving logic. Section 2 discusses a simplistic representation of drunken logic that can be modeled by Lax Logic, whereas Section 3 shows how this modeling behavior breaks down for a more precise formulation. Section 4 concludes after arguing (drunkenly!) for more investigation into this and other concerns of Chemically Assisted Reasoning.

1 Introduction to handwaving logic

Handwaving logic grew out of a concern to create better models of the way people actually use logic in the real world. Current logical systems effectively model logic in a manner acceptable to most logicians and type theorists; furthermore, the introduction of substructural logics such as linear

*This work partially supported by D's Six Pax and Dogz, and whoever supplies them with napkins.

$$\frac{\Gamma \vdash A\ true}{\Gamma \vdash A\ handwave}\ \text{INTRO} \qquad \frac{\Gamma \vdash A\ handwave}{\Gamma \vdash \text{✋}A\ true}\ \text{MODAL}$$

$$\frac{\Gamma \vdash A\ handwave}{\Gamma \vdash A \wedge B\ handwave}\ \text{EXERCISE-FOR-READER-1}$$

$$\frac{\Gamma \vdash B\ handwave}{\Gamma \vdash A \wedge B\ handwave}\ \text{EXERCISE-FOR-READER-2}$$

$$\frac{\Gamma \vdash A \vee B\ handwave \quad \Gamma, A\ true \vdash C\ handwave}{\Gamma \vdash C\ handwave}\ \text{OTHER-CASE-SIMILAR-1}$$

$$\frac{\Gamma \vdash A \vee B\ handwave \quad \Gamma, B\ true \vdash C\ handwave}{\Gamma \vdash C\ handwave}\ \text{OTHER-CASE-SIMILAR-2}$$

Figure 1: Some of the rules of handwaving logic (elimination rules are the standard ones)

logic shows promise in applying methods from proof theory to the work of robotics, A.I. and security researchers. However, current proof theoretic approaches are entirely inadequate for half of the statements made by an introductory mathematics textbook, and for even the most basic statements made by your average politician.

Handwaving logic addresses these concerns by conservatively extending standard intuitionistic logic with a *handwaving judgment* described by "*A handwave*", which is internalized in the *handwave modality* ✋. The monad admits much more powerful non-standard reasoning techniques than are generally accepted in the uptight, narrow-minded intuitionistic logic. The power and convenience of the handwave modality is evidenced by the following judgment.

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma \vdash B\ true}{\Gamma \vdash B\ handwave}\ \text{INTRO}}{\Gamma \vdash A \wedge B\ handwave}\ \text{EXERCISE-FOR-READER-2}}{\Gamma \vdash A\ handwave}\ \text{HW-AND-E1}}{\Gamma \vdash \text{✋}A\ true}\ \text{MODAL}$$

As described in [6], logic as it is used in the real world can be modeled by equating ✋A with A. Alternatively, standard techniques described in [10] can be utilized to reverse-engineer the *handwoven proof obligation* as the implicit constraint on the lax monad.

2 A brief discussion of drunken logic

As Bovik has famously noted in [2], nowhere outside of undergraduate lectures are scholars are more prone to sweeping generalizations than at the bar. Furthermore, judgments which cannot be evidenced outside of the presence of alcohol, such as the notion "I find A attractive," obviously may (in some circumstances) be proven under the constraint of drinking. Drunken logic can simple be expressed, more or less, by reinterpreting the handwave modality 👋 as the shotglass modality $\square$. In this reinterpretation, *A true* maps onto *A attractive*, while the modal judgment *A handwave* maps to *A beergoggles*. Our extensive investigations have shown that this model along with other, similar ones (such as the related game-theoretic judgement *A i'lldoitifyou'lldoit* that is vastly amplified under the $\square$ modality) are sufficient to model the vast majority of lapses in judgment under the influence of the $\square$ monad.

3 The challenge of durnken logic

Per Per Martin-Löf [7], something is true when witnessed by an *object of knowledge*, which lends itself to an obvious question of whether the truth of a proposition can be obviated by the presence of alcohol, seeing as alcohol has an clearly negative impact on one's knowledge [1]. The possibility of the analytical truth of a proposition becoming questionable under the influence is also evidenced by discussion as to whether conference submissions that can be understood while drunk are novel enough to be worth accepting.[1]

Indeed, in the above presentation of drunken logic the things that are "potentially true" are in a sense monotonically increasing; there is no provision for things that are *true and provable* while clean and sober to be revoked under the influence of alcohol. Put another way, while *A attractive* may be true under sufficient "monadic influence," it is necessarily negated if one has drunk themselves to sleep, blindness, or need of medical attention. Speaking of medical attention, Girard approaches a similar problem in his discussion of his glossary discussion MEDICINE as a problem of only being able to work with *positive information*: "the typical technique in medecine is to work only with positive information 'As far as we know, one cannot get `AIDS` by blood transfusion'" (fixed-width font in the original) [5].

[1] John Renyolds, personal communication on the Wean elevators.

We call models that must deal with models that must handle the nonmonotonic changes present in extreme modal situations *durnken logic*; in fact, drunken logic should be considered merely a special case of this more general situation. One could imagine that a constraint-based system might be able to handle blood-alcohol information such that a sequent is only considered in relation to an external constraint $\mathcal{C}$, represented as $\mathcal{C} \mid \Gamma \vdash A$. A classically-based system might be able to handle the nonconstructive truth of certain drunken judgments. Chaudhuri also demonstrates a notion of contradiction in intuitionistic linear logic such that certain contradictions can arise without being catastrophic to the overall consistency of the system [3]. A connection to linear logic would also allow us to investigate connections between the consumption of resources and the consumption of alcohol, perhaps giving a satisfactory logical justification for the truth of the "Three Tequila Proposition":

$$\mathsf{tequila} \otimes \mathsf{tequila} \otimes \mathsf{tequila} \multimap \bot$$

Most promising, perhaps, an approach based on modal or hybrid logic could internalize states of drunkenness within a Kripke model - this would potentially generalize to other chemical modalities, such as the observation of certain researches in this area that application of their particular modal operator was akin to transportation "out of this world."[2]

4 Conclusion: Chemically Assisted Reasoning

It is not our intention in this paper to solve all (or any) of the problems we present; rather, it is to propose various approaches to this rich research area that has previously been explored only in the most ad-hoc manner. In general, we think that the opportunities for generalization presented in the previous section point to rich opportunities awaiting researchers in the field of *Chemically Assisted Reasoning* (or "CAR" - but don't drink and drive). Reasearch areas far afield from proof theory, such as work on *regret minimization algorithms*, also have obvious applications to the logic of drinking, and to chemically assisted reasoning in general [9].

[2] Anonymous personal communication, Terrace Club, Princeton University, Fall 2004.

References

[1] Harry Q. Bovik. Programming under the influence: a comparative approach. *Mathematical Structures will Drink You Under the Table*, 7(7), 1977.

[2] Harry Q. Bovik. My life in a nutshell, if by nutshell you mean bottle of cheap whiskey. In *13th International Symposium on Principles and Practice of Declarative Declarations*, February 1985.

[3] Kaustuv Chaudhuri. *The Focused Inverse Method for Linear Logic*. PhD thesis, Carnegie Mellon University, December 2006.

[4] Matt Fairtlough and Michael Mendler. Propositional lax logic. *Information and Computation*, 137(1):1–33, August 1997.

[5] Jean-Yves Girard. Locus Solum: From the rules of logic to the logic of rules. *Mathematical Structures in Computer Science*, 11:301–506, 2001.

[6] Neel Krishnaswami, Rob Simmons, and Carsten Varming. Handwaving logic. Journal of the Eighth Floor Whiteboard, December 8, 2006. Possibly erased.

[7] Per Martin-Löf. On the meanings of the logical constants and the justifications of the logical laws. *Nordic Journal of Philosophical Logic*, 1(1):11–60, 1996.

[8] Frank Pfenning and Rowan Davies. A judgmental reconstruction of modal logic. *Mathematical Structures in Computer Science*, 11(4):511–540, 2001.

[9] Aaron Roth. Personal communication, Wean Hall 4120.

[10] Privately circulated memoir of the Slovenian Philological Society, 1947.

Morality, amorality, and truth irrelevance in a nihilistic type theory (abstract)

Ezra Broemeling
Carnegie Mellon University

March 6, 2007

The rules of logic are fundamentally arbitrary. How does one choose between dependent and second-order quantifiers? Between weak and strong negation? Between linear and bunched implication? Arguments one way or the other are simply unjustified expressions of *faith.* In this paper, I propose new foundations for logic based on a *nihilistic type theory,* which has no introduction rules and no elimination rules. Nihilistic type theory (NTT) thus defines a logic completely free of dogma, *beyond* truth and falsehood—but which is nonetheless extremely powerful: I sketch soundness and completeness of NTT with respect to the extended calculus of constructions. Soundness is shown by giving a fully constructive, type-preserving translation taking NTT terms to ECC terms. Completeness is more difficult, but can be proven using the technical machinery of *truth irrelevance.* Finally, I discuss applications of the nihilistic conditional $A \nrightarrow B$, used to express the proposition that A implies B, but it doesn't matter, because in the end we're all going to die anyways.

Bipolar Logic with Lithium*

Daniel K. Lee

Carnegie Mellon University

Abstract

In prior work, we explored Bipolar logic, which contains two modalities, a *manic* modality under which any number of intuitionistically unsound propositions can be proven and a *depressed* modality under which only the most trivial propositions could be proven. Despite the promising applications of bipolar logic in the areas of generating Dutch Post-Impressionist graphs, M.A.C.H.O. expatriate ciphers, and Grunge compositions proof search in bipolar logic is made exceedingly difficult by the unpredictable and dramatic shifts between the *manic* phase and the *depressed* phase. Even the most successful applications of bipolar logic are difficult to assess, because these most successful programs are so unstable they tend to suddenly self-destruct with little hope of recovering the old code.

We propose a Linear Bipolar Logic with Lithium. The presence of lithium resources controls for the sudden shifts between *manic* and *depressed* phases, making proof search more tractable. However, we leave as an open question whether Linear Bipolar Logic with Lithium is as expressive as the traditional presentation of Bipolar Logic.

*This work is partially supported by the National Science Foundation under a Graduate Research Fellowship and D's Six Pax and Dogz.

The Letter before Lambda is Hat: A Reconstruction of Church's Hat Calculus

Akiva Leffert
Cranberry-Melancholy University

SIGBOVIK 2007

Abstract: We present a reconstruction of Alonzo Church's Hat Calculus based on notes discovered under a book shelf. We present evidence that this system was the precursor of the λ-calculus. We then describe the system in full detail. We prove a lack of progress theorem. Finally, we prove an undecidability result by reductio ad absurdem.

Keywords: Computability, Hats

1 Introduction

It is said by those with too much imagination that the λ-calclus sprang fully formed from the head of Alonzo Church and that as he laid the α, β, and η rules on paper, choruses of angels sang Hallelujahs. Those with less imagination and perhaps more wit realize, as Edison said, that genius is hard work and makes one sweaty. Indeed, even the character λ, namesake of said calculus, was not in Church's original work. The usual story is that he borrowed notation from Russell and Whitehead's Principia Mathematica, which used a circumflex over variables to mark abstractions[1]. Church used this early on, for example, writing the identity function as: $\hat{x}.x$. However, due to inferior typesetting technology, the circumflex shifted from above the variable to its left like so: $\hat{}x.x$. This appearance of $\hat{}$ resembles a capital lambda, Λ, which caused some other typesetter, mind no doubt dulled by too much exposure to hot-lead, to use the lower-case λ we know and love. Thus, except for a typographical accident, the λ-calculus would be known as the circumflex-calculus or, more succinctly and colloquially, the hat-calculus.

This sepia-tinged tale of typography makes for a good story to tell little freshlings flush with curiosity about the λ-calculus and the Entscheidungsproblem[3][6].

Harry Bovik, demonstrating his diverse talents, actually made a short film about this[2] which was well received[5]. However, it is wrong in one important detail. Old notes of Church's, recently discovered stuffed under a shelf in the Princeton University Library suggest that this notation was inspired, not by Russell and Whitehead, but by an earlier system which Church sketched out and described in those notes. The account of this discovery can be found in [7]. This system of computation contained more literal hat symbols - see Figure 1. In the remainder of this paper, we describe this system, the Hat Calculus, and sketch a proof of the undecidability of the Down-Feather Problem by reduction from the Halting Problem.

2 Hat Calculus Syntax and Semantics

The complete syntax of the Hat Calculus appears in Figure 2. This system is considerably more, umm, baroque than the λ-calculus. This suggests that Church learned a great deal from the development of this system, abandoning it due to its complexity rather than any inherent computational weakness of the system. Indeed, we later show that this system is Turing-complete.

Definition 2.1 (Up Feather): The symbol is an *up-feather*.

Definition 2.2 (Down Feather): The symbol is a *down-feather*.

Definition 2.3 (Banded Hat): A hat can be combined with a band to create a *banded hat*. For example, a can be combined with a to create . The latter is a banded hat. For the purposes of clarification we may, but probably won't, occasionally refer to hats without bands as *naked hats*.

Definition 2.4 (Banded Feathered Hat): Hats with bands can be combined with feathers to create *banded feathered hats*. For example, is a down feather banded hat as is .

Definition 2.5 (Action Card): All cards in the Hat-Calculus are the same except for the *action cards*: $\boxed{\spadesuit_7}$, the $\boxed{\heartsuit_8}$, the $\boxed{\heartsuit_9}$, $\boxed{\heartsuit_{10}}$.

Definition 2.6 (Inaction Card): A card which is not an action card is an *inaction card*.

Definition 2.7 (n-Carded Banded Hat): Each banded hat is actually an *n-carded banded hat* for some n. An n-carded banded hat is a hat with n cards associated with it. An uncarded hat is just the degenerate case when n is zero. Note that while there are fifty-two (four times thirteen (two times

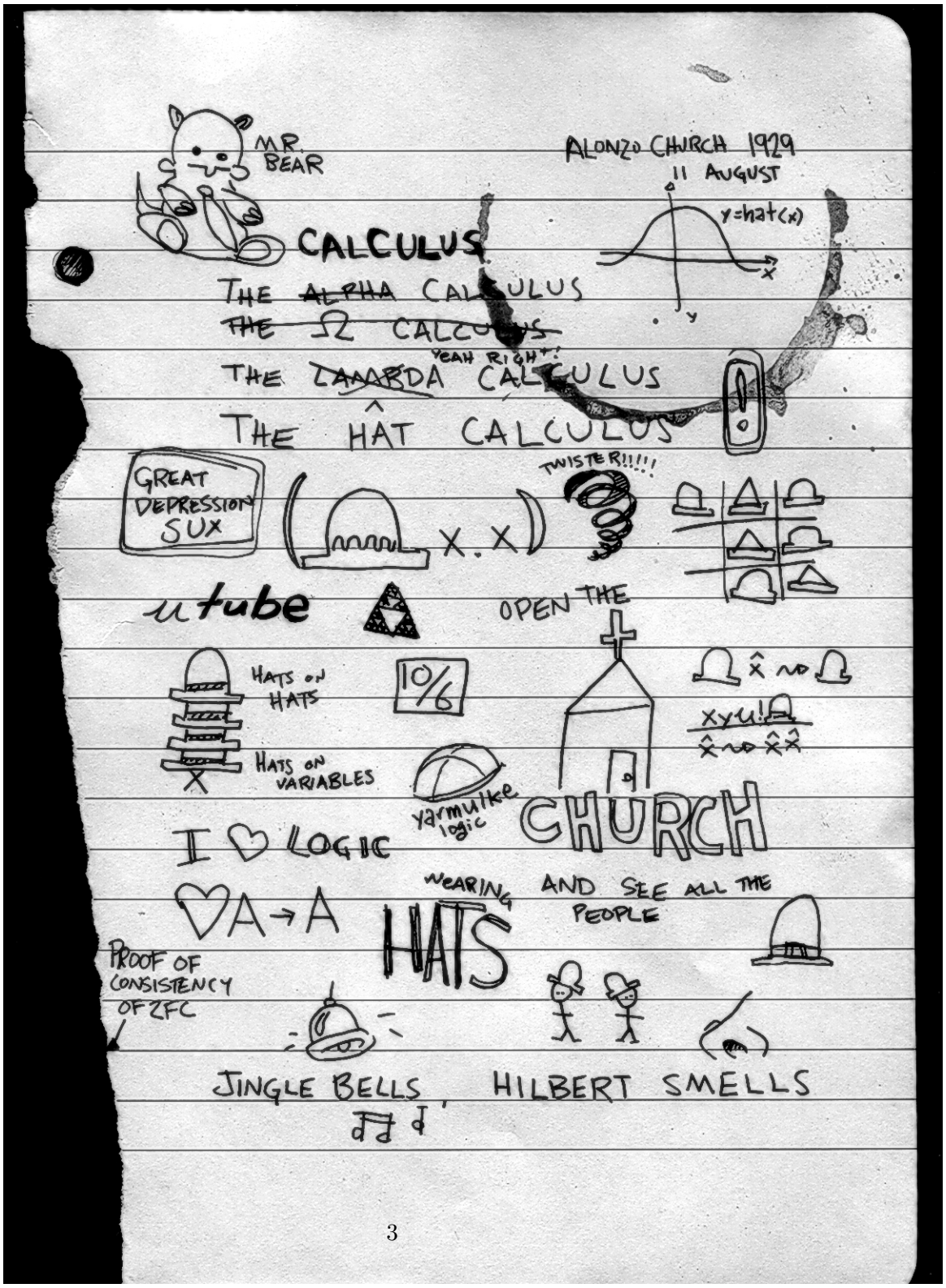


Figure 1: Excerpt From The Lost Notebook of Alonzo Church

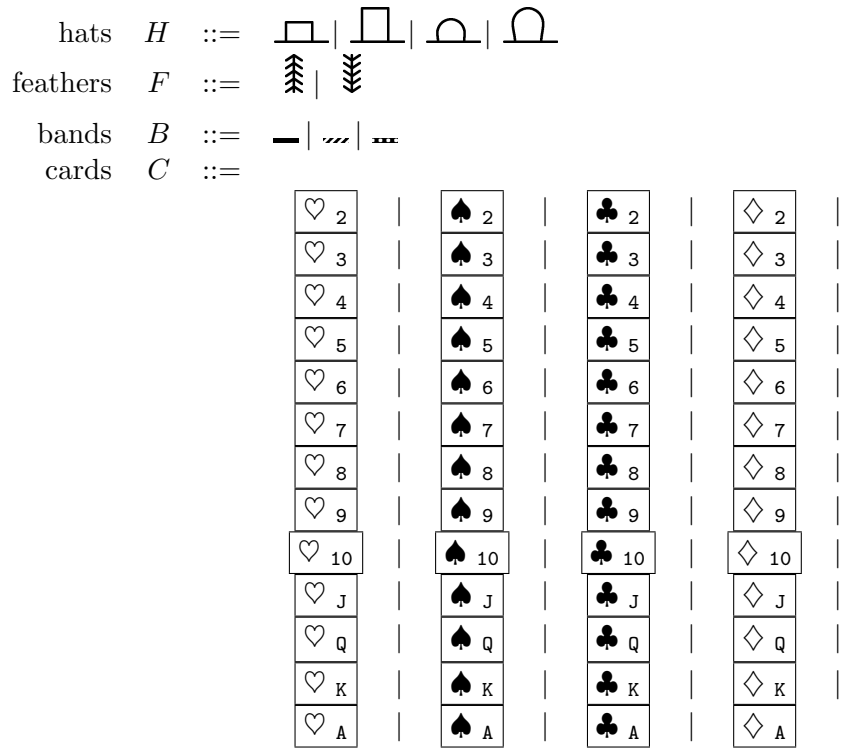
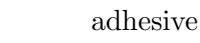
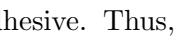


Figure 2: Hat-Calculus Syntax

two times thirteen)) cards, only four of the cards are distinguished by the semantics of the language, the action cards. We suspect that Church was perhaps not at his best when designing this system.

Definition 2.8 (Final State): A hat-calculus expression is considered final if all of the feathers are down feathers.

The process of computation is the process of attaching bands to hats, feathers and cards to banded-hats, and stacking hats on other hats. As in the lamba-calculus, juxtoposition is application, in this case, application of adhesive. Thus,  steps to  . Unfortunately, this convenient combination notation doesn't work as well as we add cards and feathers to hats. Thus, we use the notation $B(H, F, [C_1, \ldots, C_n])$ to represent a completely applied n-carded banded hat. If we wished to combine an n-carded banded hat with another card, say, the King of Hearts, $\boxed{\heartsuit_{\mathtt{K}}}$, we would write this like so: $B(H, F, [C_1, \ldots, C_n])\boxed{\heartsuit_{\mathtt{K}}}$. This would step to $B(H, F, [C_1, \ldots, C_n, \boxed{\heartsuit_{\mathtt{K}}}])$. Note that combining action cards has a different effect discussed later.

Hats can be stacked. If two hats are juxtaposed we combine them into a stack. Stacks of hats can also be stacked in this manner. It is not possible to combine a stack of hats with a single hat in this manner. The application of a band to a stack of hats has the effect of applying that band to all of the hats. If the band of a hat is replaced it loses all of its cards and feathers.

2.1 Action Cards

Combining the $\boxed{\spadesuit_{\mathtt{7}}}$ card with a hat or stack of hats causes all of the feathers to flip - i.e. all down feathers become up feathers and vice versa.

Combining the $\boxed{\heartsuit_{\mathtt{8}}}$ card with a stack of hats removes all of the hats from the top and bottom until a hat is reached with a down-feather.

Combining the $\boxed{\heartsuit_{\mathtt{9}}}$ with a stack of hats duplicates the stack.

The $\boxed{\heartsuit_{\mathtt{10}}}$ is the *ungluer*. It pulls all of the feathers, bands, and cards off of a hat. It also creates a new action card at the end of the expression. Which particular action card is chosen is non-deterministic.

3 Results

Definition 3.1 (Stuck): An expression of the Hat-Calculus is *stuck* if it cannot step and is not a final state.

Theorem 3.1 (Lack of Progress): There exists a stuck state. Proof: The expression  cannot step, but is not final.

Definition 3.2 (Down-Feather Problem): The down-feather problem asks whether a given Hat-Calculus expression will reduce to a stuck state.

Theorem 3.2 (Universality): We're pretty sure it's Turing complete. It's got a queue or something.

Theorem 3.3 (Undecidability): The Down-Feather problem is undecidable. Like we said, it's probably Turing complete.

4 Conclusion

The Hat-Calculus was developed by Alonzo Church before the λ-calculus. It is a Turing-complete language of computation with a rather ungainly syntax. Actually, it's unarguably nonsense[4]. Fortunately, Church later developed the λ-calculus, which isn't crap (we hope).

References

[1] Henk Barendregt. The impact of the lambda calculus on logic and computer science. *Bulletin of Symbolic Logic*, 3(3):181–215, 1997.

[2] Harry Bovik. Lambda-calculus: The feature film extravangza. Feature Film.

[3] Alonzo Church. A note on the Entscheidungsproblem. *Journal of Symbolic Logic*, 1:40–41, 1936.

[4] Jean-Yves Girard. Locus solum: From the rules of logic to the logic of rules. *Mathematical. Structures in Comp. Sci.*, 11(3):301–506, 2001.

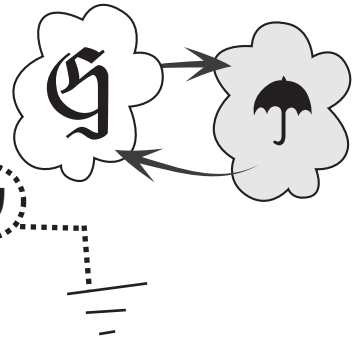
[5] Fred Hacker. Harry Bovik shouldn't be let near a camera. Letter to the Editor.

[6] Richard Karp. The Entscheidungsproblem is probably NP-complete. Private communication in an elevator.

[7] William Lovas and Tom Murphy VII. The hidden finds of janitorial work. *Proceedings of Found Stuff Symposium*, 9(1):1299–1578, 2005.

A Diagrammatic Notation for F;

Jason Reed
Carnegie Mellon U



Abstract

Category Theory, String Theory, Knot Theory, Graph Theory, Proof Nets, Feynman Diagrams, and Penrose's tensor contraction notation: all too often has been demonstrated the value of graphical and diagrammatic reasoning in advanced mathematics. By replacing incomprehensible piles of linear syntax with equally incomprehensible piles of funny squiggles and wildly pointing arrows, formal diagrams have enhanced the visual appeal of written work while maintaining, or in some cases improving on the status quo of outsider-repelling intimidation.

1 Introduction

The vast majority of research in formal systems is performed on a two-dimensional page (with the exception of the burgeoning field of Virtual Real Analysis, which requires special goggles and red-green differential operators) and yet this spatiality is often wasted by intrinsically one-dimensional notation. We aim to fix this problem by introducing a clear and precise two-dimensional notation for the foundations of mathematics and logic. In the sequel, we try to avoid any use of linear ordered multi-sets of character-based information units (i.e., ordinary running text) except when strictly necessary.

2 Syntax

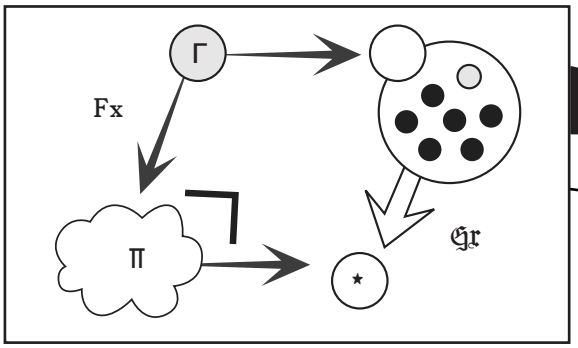


Fig 1. Neither a pullback nor a pushout be

3 Type Theory

Our typing rules are a healthy part of a sound, complete breakfast.

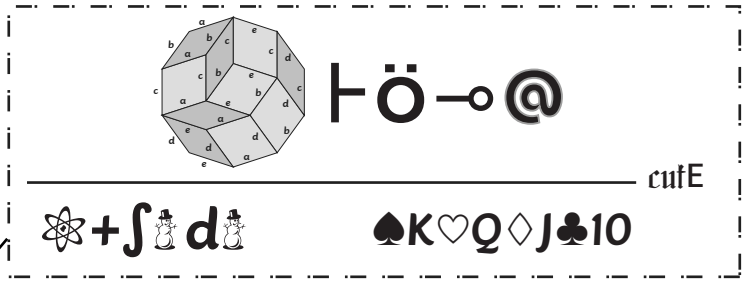
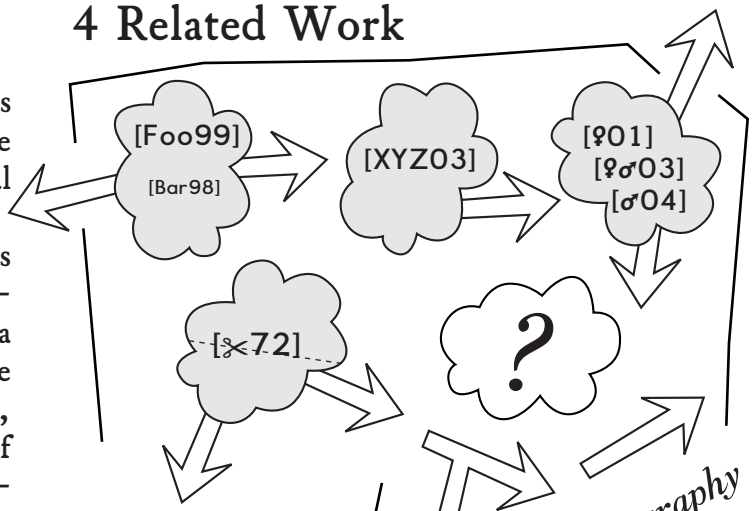


Fig 2. 100 per cent RDA of heavy metal umlauts

4 Related Work



5 Conclusion

Fig 3. Cryptobibliography

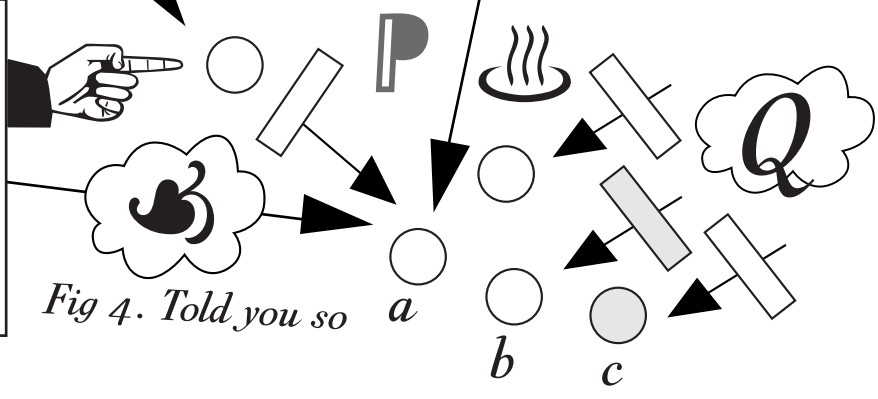


Fig 4. Told you so

First Impressions

A Modal Logic Designed Specifically to Confuse Undergraduates

Matthew Kehrt
University of Washington
mkehrt@cs.washington.edu

The Idiran Empire
OUTER SPACE
enslavethegalaxy@hotmail.com

Abstract

beepbeep

1. Introduction

One major attraction of modern type theories lies in their elegant use of pretty symbols. What modern computer scientist could not but be smitten by the simple beauty of such gems as

$$\Gamma \vdash e{:}\tau$$

or even

$$\Gamma \vdash M{:}A$$

Hell, even funny symbols such as those used in

$$\Gamma; \Delta \vdash M{:}A \multimap B$$

are pretty ok.

However, these beautiful strings of symbols serve a deeper purpose than mere aesthetics. They also allows us effectively hide very powerful ideas in a mass confusing typography and terminology. For example, it is rare that one needs to actually describe what one is working on when it includes such terms as "pointwise subkinding." [2, 3]

With rising education and theoretical programming language ideas become more accepted by a broader audience, it is becoming more and more necessary to increase the complexity of terminological and typographical conventions if any actual work is to be accomplished [1].

With this in mind, we present FIRST IMPRESSIONS, a modal logic designed specifically to confuse the unitiated.

Propositions	P	$::=$	$P_1 \vee P_2 \mid P_1 \wedge P_2$
		$\mid$	$P_1 \heartsuit P_2 \mid MP$
Atomic Propositions	M	$::=$	$A, B \mid \triangle M \mid \Box M$
		$\mid$	$\cdots \mid \bigcirc M$

Figure 1. Some symbols

2. Confusing Overview

Propositions in FIRST IMPRESSIONS consist of base propositions prepended by a series of alethic modalities. Base propositions are really whatever one feels like: they do not actually matter at all. We're not actually going to do anything with this logic other than talk about it. So, for instance, one could have a series of atomic propositions, $A, B, ...$, and some binary connectives, $\vee, \wedge, \rightarrow, \propto, ©$.

Modalities consist of the set of regular polygons. They are distinguished by how many sides they have. Any polygon with more than nineteen sides is considered to be a circle, which has nineteen sides. In notation, these polygons may be written with a dot in the middle. This has no meaning, but serves merely to multiply notation.

A string of modalities can be equivalent to another string of modalities. A string of modalities followed by a base proposition A is true iff an equivalent string of propositions followed by A is true. However, the exact rules determining the provability of the truth of a proposition are unclear. Therefore, it is necessary to give heuristics for determining the if two strings of modalities are equivalent. In practice, we find the most useful way of determing this is to sum the number of sides of polygons in a string and take that number modulo twelve. If two strings yield the same number, they might be equivalent.

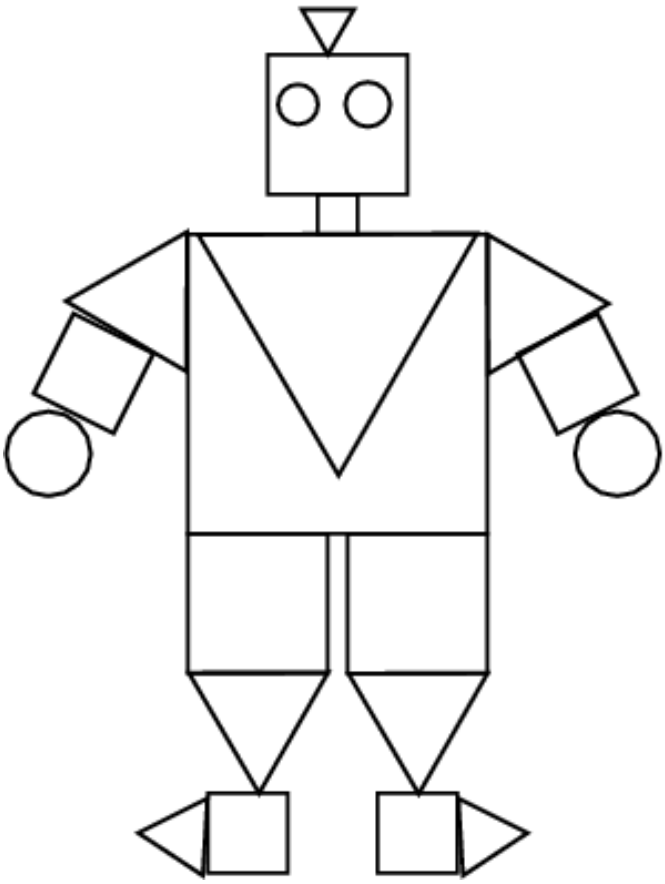


Figure 2. A Robot

For example, it is pretty likely that a cicle (nineteen sides) is equivalent to a heptagon, as $19 \equiv 7 \mod 12$.

We find that discussion of these rules leads to endless confusion in those who are unaware of the subtlties of modal logic. Even those who have much experience, when faced with a pentadecagonal modality, tend to give up quickly.

3. Metatheory

As we do not actually have any judgments, inference rules or even any particularly well specified syntax, metatheory for this logic consists mostly of vague statements about possible future work. This is a striking return to the philosophical underpinnings of modern logic, and we hope to someday treat this subject in full.

4. Use

One of the primary uses of this logic is for drawing pictures. Freed from the tradional constraints of simple boxes, diamonds and possibly circles, we have a near unlimited palette of shapes from which to choose. However, we still require these shapes to be regualar which makes more complicated images difficult to compose. We find that, given these limits, houses and boxy robots (Figure 2) are some of the most easiest images to produce.

Acknowledgments

Thanks to Akiva Leffert for being confused, too.

References

[1] J.-Y. Girard. Locus solum: from the rules of logic to the logic of rules. *Mathematical structures in computer science*, pages 11:301–506, 2001.

[2] M. A. Kehrt and A. Kehrt. Personal communication, November 2005.

[3] W. Lovas and K. Crary. Hot Compilation: Elaboration. Homework assignment, November 2005.

Track II:

(Photo-)Realistic Applications

Towards an Evolutionary Next Generation Avian Carrier Internet Architecture

Cary Rebecca Pidgin

February 28, 2007

Categories and Subject Descriptors

C.2.1. [**Computer Communications and Networks**]: Network Architecture and Design

General Terms

Avian Carriers, Internets, Tubes, Internet Architecture

1 Introduction

In the last 10 years, there have been many design proposals for a next generation Internet architecture intended to improve the performance, security, and availability of today's Internet. While these proposals have mostly leveraged recent advances in network hardware and traffic trends, there has been a dearth of research looking at a new Internet architecture based on CPIP (Carrier Pigeon Internet Protocol)[2]. We believe it is important to fully explore the design space of architecture possibilities to better inform the community about what should go into a new architecture.

Avian-based protocols have until recently been considered flights of fancy. In this paper we discuss how to extend the principles of CPIP (Carrier Pigeon Internet Protocol) to handle a variety of modern applications. In particular, we show how our Avian Carrier Internet (ACI) architecture supports an evolutionary model that can adapt with the changing communication models of Internet traffic. We begin with a short review of CPIP and some basic additions to the protocol and "birdware" to modernize the technology for today's environment. Next, we explain ACI's methods of providing an evolutionary framework for future Internet applications. We then discuss the potential drawbacks of implementing such a scheme. Eventually, we close with a brief discussion of related work and potential avenues for further research in this area.

2 Birdware Modifications

After months of research, we discovered that the main problem with RFC1149 and RFC2549 [3] is the use of paper scrolls for data storage. Such a system does not take advantage of the improving capacity of flash drives. Our modification is to replace the paper scroll data storage technology with the latest in NAND flash technologies. Furthermore, in concert with the avian trend, we use duck tape instead of duct tape to secure the drive to the birdware's legs[1]. In order to support an end-to-end model of security, the data on the flash disk can be encrypted by one-time-pad technology. We use paper scroll data storage technology to store the one-time-pad on the other leg for nearly unbreakable security guarantees.

3 Evolutionary Model

Perhaps the most important contribution of this paper is the unintuitive melding of science and technology. We leverage off of recent Darwinian models of evolution, though we are open to other models of

[1] If the birdware is of the duck species, we may be able to avoid such methods.

evolution should they be discovered. In our architecture, we use avian breeders to selectively choose for species traits that are conducive to the architecture. The end goal of avian breeders is to produce fast-flying, super-strong avian carriers. These traits are necessary to provide low-latency, high-bandwidth characteristics. The intelligence of the avian carriers must necessarily be sufficient to prevent forwarding loops from occurring in Super Wide Area Network (SWAN) environments.

As genetics research continues to flourish, we envision a transition to genetically-created species to provide a quicker turnaround time for avian carrier modifications. For example, we may use the hawk's quick diving speeds along with the pterodactyl's enormous size as one possible avian carrier species. We also envision the possibility of a hummingbird-cheetah species to support real-time applications such as Voice-over-Internet-Peregrine and pigeon-based CooTube video content delivery.

4 Issues of Packet Loss

ACI continues to push the 'best-effort' model of data delivery on the Internet as the original RFCs intended. As a result, packet loss is often inevitable. In particular, ACI is susceptible to normal types of packet loss expected with an avian-based datagram system, such as glass windows, sky-blue painted buildings, redneck hunters, and avian bird flu pandemics. We expect that with time, avian carriers can be scientifically engineered to be resistant to such sources of failure, although people seem to really like glass and we have no current solutions to this problem (perhaps human computation methods are necessary).

One open problem is that of Denial of Service attacks. Figure 1 shows an example scenario [1] that could result when an attacker tries to packet flood a destination comprised of children. One possible solution is to arm every human with a device capable of putting up a wall of fire, or what we term a 'firewall,' to defend against these attacks.

Figure 1: Potential Damage by DoS Attack in ACI

5 Future Work and Conclusion

While ACI provides an architecture for an evolutionary next-generation Internet architecture, its success depends on the ability of genetics research to combine hummingbirds with cheetahs. Given the slow process of legislation for avian stem-cell research, we believe there is merit in leveraging steroids to help provide incremental benefits to avian-carrier networks. Finally, we are looking at ways to improve latency with wormhole routing using our EBGW (Early Bird Gets the Wormhole) protocol.

Given our initial goal of exploring the design space of Internet architectures, we believe ACI stands out as a contrarian design that should improve networking research around the world. Our hope is that other Internet architecture proposals will continue to build off of our design for make benefit future generations.

References

[1] A. Hitchcock. The Birds, 1963.

[2] D. Waitzman. A Standard for the Transmission of IP Datagrams on Avian Carriers. RFC 1149 (Unrecommended Standard), April 1990.

[3] D. Waitzman. IP over Avian Carriers with Quality of Service. RFC 2549 (Unrecommended Standard), April 1999.

Compacting, Composting Garbage Collection

Jake Donham, Carnegie Mellon University

Abstract

Garbage collection is vital for programmer efficiency, but hides the societal costs of rampant waste of data. We present a means to reduce the negative externalities of garbage collection through natural mechanisms of waste reprocessing. We demonstrate a 38% reduction in allocation on a suite of ML benchmarks, as well as a 47% increase in the growth of plants fertilized with the rich, loamy byproduct.

Keywords: bioengineering, programming languages, dung

1 Introduction

While great strides have been made in recent years in improving the space and time overhead of garbage collection, as well as its realtime behavior, little attention has been paid to the environmental consequences of the style of "disposable programming" which garbage collection encourages, in which large numbers of data structures are created, only to be thrown away almost immediately. Such abandoned data puts a strain on the waste management capabilities of a typical software system as well as the social context in which the system operates. Moreover, the cultural impact of this style of programming is to encourage the wasteful discarding of perfectly good data which could be repaired and put back into service, thereby stimulating a vibrant economy of small-scale local artisans along the lines of neighborhood tailors and shoemakers.

To address one aspect of this deficiency of existing methods of garbage collection, we propose *compacting, composting* garbage collection. The core idea is that unreachable garbage, once identified by a collection algorithm, should not simply be discarded, but should be repaired and reused if possible, and otherwise encouraged to decay into a nutrient-rich soil. This method is completely orthogonal to traditional garbage collection algorithms, and in fact we have implemented it in a family of collectors including stop-and-copy, mark-and-sweep, clean-and-jerk, sit-and-spin, and a hybrid transcendental, intergenerational, centrifugal collector.

2 The algorithm

The compacting, composting collector works by segregating the heap into several *piles*, corresponding to garbage objects of different ages. This is dual to the generational approach of segregating live objects of different ages, and rests on a dual generational hypothesis that "dead objects stay dead", or, equivalently "objects aren't getting any younger". The idea is that as dead objects age and decompose, they are moved to successively older piles, which they share with dead objects of roughly the same age. This serves to isolate the stinkiest parts of the heap, as well as to produce, in the oldest generation, a uniformly decayed pile which can be scooped out and used to fertilize future computations.

There are two additional phases of the algorithm: repair, in which discarded objects which need only a bit of work with needle and thread, or some common white glue, or a little oiling, are fixed, cleaned up, and put back into service; and compaction, in which garbage piles are pitchforked to break up clumps and then tamped down with a shovel.

As an optimization, we keep a special pile for inorganic objects which do not decay on the same timescale as typical objects. If objects on this pile cannot be repaired and reused, they may be taken to the dump or left at the curb for pickup. Also, we encourage quicker composting by seeding piles with beneficient bacteria and worms.

3 Our testbed

We have built and measured our collector in the SILT compiler for Standard ML. SILT (Structured Intermediate Language, Too) is a structure-preserving compiler, which compiles programs by successively transforming them into a series of structured intermediate languages, such as SOIL (Structured Op-

erational Intermediate Language) and DIRT (Direct Intermediate RepresenTation). The SILT approach provides large benefits in the form of increased compiler correctness, additional opportunities for optimization, and an organic, holistic, centered user experience, man. Can you dig it?

Test programs included a variety of climate-modelling, SETI-at-Home, and non-violent video game workloads. We attempted to test the collector with a nuclear weapon yield computation and a finance package but found that these programs made assumptions incompatible with our method.

4 Results

In side-by-side comparisons with a standard collector, we found that overall 38% of objects could be repaired and reused across the benchmark suite. The high-quality compost resulting from the final pile was sold to local farms at an average price of $112 per ton.

We have omitted detailed graphs in an effort to cut down on paper.

5 Related work

The most similar work to our is contained in Davis' thesis [1], which presents the design of a language (called Lollipop) in which you need only say what you wish to be done, including a memory management subsystem that picks up after the programmer and puts away his or her objects neatly. However, Davis does not provide an implementation.

In a tour-de-force of analysis, Smith et al. [4] derive a space bound on waste produced by a herd of Holsteins on a farm in Iowa. Jonas [2] proves a theoretical limit on the efficacy of biocomputational methods, by a reduction to graph-coloring. Murphy [3] evaluates the cache behavior of locally-grown objects.

6 Future directions

While our method is effective in reducing the waste produced by a program, purely through modification of the garbage collector, the larger problem of waste management must be addressed further upstream, at the point that the garbage is created. We are therefore re-evaluating methods of explicit memory management, in which a programmer who knows that a particular object can be re-used adds it to a *free list*, from which future allocations can be made. Furthermore, if the programmer knows that an object is no longer needed, he or she may explicitly free it for recycling, rather than allowing garbage to accumulate.

Over the long term, we hope to encourage programmers to move away from comfortable, yet environmentally-suspect languages such as ML, which provide automatic memory management and abstraction facilities that hide the true origin of data (a form of Marxist commodity fetishism), and return to the honest, handcrafted code of their forefathers.

7 Conclusion

We have shown that compacting, composting garbage collection is both feasible and useful. We hope that this contribution will help bring about a new age of low-impact programming and green systems.

References

[1] C. Davis. *Passive-Aggressive Programming*. PhD thesis, Department of Computer Science, Cranberry Melon University, 2001.

[2] J. Jonas. Crop rotation is NP-complete. In *International Conference on Computational Agriculture*, pages 83–91, Braga, Portugal, 2003.

[3] T. Murphy VII. Exploiting data locality: Fresh bytes and community supported agriculture. *Journal of Environmental Semantics*, pages 117–127, 1998.

[4] J. Smith and Z. Biddleworth. Free-range analysis and abstract irrigation. In *Formal Methods in Farming*, pages 190–197, Ames, Iowa, USA, 1985.

C Dereferenced

Akiva Leffert

March 20, 2007

Abstract

We present the *C language. Named in the spirit of C++ and C#, *C is what happens when the C language is dereferenced. Unfortunately, dereferencing C results in a SEGFAULT. *C adds a variety of syntactic features to C, all of which will crash your machine and destroy any resident data. We also briefly investigated the &C language only to realize that there was only one real C reference - Kernighan & Ritchie's *The C Programming Language*.

The GUnit Testing Harness: Achieving Source Code Street Cred

Nels E. Beckman

***Abstract*— In any large software organization, the street cred of source code is of principle concern. Often-times we find that a given code base can talk a mean game, but when time comes to "throw down," that code base is nowhere to be seen. While intuitively we as software developers come to gain a sense of which code bases are trust-worthy (or, "will ball 'till they fall"), some systematic method for measuring and reporting this is necessary. In this paper we present the GUnit testing harness (pronounced, "Gee Unit"), which does just that.**

***Index Terms*— Software Engineering, Credibility Improvement, Street Life, Hennessy, Testing, Java**

I. INTRODUCTION

Slingin' code is a ride-or-die way of life, where every day it's just another homicide. It is important to stay TRU to tha game, or you risk being played for a chump. But the question is, how does one go about evaluating (and eventually increasing) the TRU-ness of the code that they sling? This is especially important at large software organizations where code often becomes soft from spending too much time in cushy source-code repositories.

In this paper we present the GUnit Testing harness, a tool created explicitly for the purposes of evaluating the street cred of a code base. GUnit is the fruit of a rare cross-discplinary collaboration. Researchers from Carnegie Mellon University's Institute for Software Research have joined together with the Interscope Records' Gorilla Unit, lead by world-renown scholar Dr. Curtis James Jackson III, who goes by the pseudonym 50-Cent.

While GUnit currently only works on Java code bases (an inherrently 'soft' programming language, when compared with other, 'harder' languages such a C and z80 assembly [5]) we believe that our underlying street cred evaluation methodology can easily be extended to other languages.

This paper proceeds as follows. In Section II, we discuss our underlying methodology. How does one take any piece of source code and evaluate its worth in the inner-city? In Section III, we describe implementation and usage of the GUnit tool. Most helpful to the end-user is Section III-A, which describes the classifications into which we place evaluated code. Section IV discusses the evaluation of GUnit and ultimately, and mercifully, Section VI concludes.

II. METHODOLOGY

The GUnit testing harness allows Java developers to build streed cred tests, and then automatically run and evaluate their results at the touch of a button. Our methods are based on earlier work by Clifford Smith, AKA the Method Man [6]. These methods are robust to programmer error and scale to large code bases as well as large quantities of Cristal. Unfortunately, these methods are proprietary and cannot be described in full technical detail.

N. Beckman is an average to below-average PhD student in the Institute for Software Research, *the* most thugged-out department in the School of Computer Science, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213 nbeckman@cs.cmu.edu

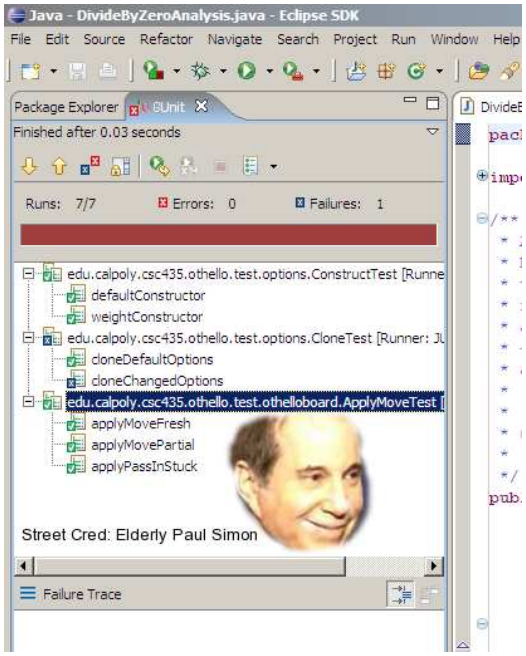


Fig. 1. Screenshot of GUnit: This source code has a street cred rating of "Elderly Paul Simon."

III. THE GUnit TOOL

The GUnit street cred testing harness has been implemented as an Eclipse plug-in [1]. Eclipse is gradually replacing Emacs as the text-editor-that-acts-like-an-operating-system of choice for modern software developers.

A. Street Cred Strata

In order to better convey the the end-user programmer the actual level of street cred of the code base under test, we have developed the following scientific classification scheme. It includes ten discreet levels, listed here in order from furthest to nearest to the street. We also briefly describe each these gradations.

1) **Clay Aiken**—At the lowest end of the street cred spectrum is the rating of "Clay Aiken." Clay's Wikipedia

Fig. 2. Hpnotiq is a 35 proof French fruit liqueur made from vodka, cognac, and tropical fruit juices [7]. Sounds good to me!

entry describes him as "the most successful second-place finisher" in the history of American Idol. If that's not a dubious destinction for your software, then what is?

2) **CMU Graduate Student**—Being a graduate student at Carnegie Mellon University doesn't get you very far; it won't get you into the club, and it won't get your court-side seats. Sometimes the truth hurts.
3) **Elderly Paul Simon**—Most thuggish software doesn't live to see 30. Elderly Paul Simon shows us why this is a good thing. Current best-practices state that one should stay young, fly, and flashy 'till the day that one dies.
4) **P-Diddy**—This is a good indication that your source code was relatively well respected during the past, possibly because it played some pivotal role in the career of a more successful product. But when the more successful product was killed in a drive-by shooting in L.A., everyone kind of realized that your product was more about marketing than quality. Then your software did a cover of a Led Zepplin song for the Godzilla remake, which only made things worse...
5) **Young Paul Simon**—Have you ever see that movie *Almost Famous*? He *always* had big pupils...
6) **Hpnotiq**—Just like the liquor of the same name, your source code is most frequently seen at the clubs, in the VIP section. If you seem him out, remember: Don't start no stuff, won't be no stuff.
7) **Chry$ler 300, with Navigation**—Seriously, that car is dope. It's kinda like the cars that the bad guys drove on Batman: The Animated Series.
8) **Ja Rule**—If you feel the need to scream "Holla, Holla!" when your code base receives this rating, that is a perfectly acceptable reaction. If only it weren't for that memory leak, the software equivalent of a J-Lo collaboration, you'd have more respect.
9) **Dead Prez**—While your source code is not, strictly, as popular as some of the other big names on this list, it is authentic and has the respect of its peers, due to a prolonged life spend on the south side of Chicago, and its refusal to sell out. The authors themselves said it best in their seminal work [2]: *That's why I'm in the dojo, not just for the video. Really though, we really got beef with the popo. Never know when they gonna put you in a choke hold.*
10) **Hova**—The Michaelangelo of flow, your source code paints pictures with poems. When source code achieves a street cred rating of "Hova," this indicates and extremely high degree of credibility. At the same time, this rating indicates that a given piece of source code has achieved a rare feat, that of becoming commercially successful while still maintaining its reputation of being true to its roots. Much like Jay-Z, the J-Hova of hip hop, a code base achieving this rating has flow that is religious.

IV. Results of Tool Deployment

We attempted to try out our tool on a user base of approximately 500 Java developers. However, due to the fact that we were constantly screaming, "Wu Tang!" and, "Don't bring those weak generics into my house!" most of the developers dropped out. Therefore our current results are inconclusive.

V. Related Work

A. CMMI

Currently, the most well-known metric of street credibility in existence is the Stuntin' Engineers' Institute's Crunkability/Make Money Index (CMMI) [3]. CMMI has long been the de-facto standard hype metric for military and government software contractors. As its name suggests, the CMMI ranks the performance of a software development organization in two specific area: Its "crunkability" (its ability to have fun, drink, and spit game at the ladies) and its ability to "Make Money" (get paid by any means necessary). Based on these qualifications, a software organization is given a rank from one, being the fakest, to five, being the realist. The primary difficulty with the CMMI system is the large amount of overhead necessary in having an organization certified. While this is appropriate for large government contracters where the crunk-ness of the entire nation is at stake, for smaller, more agile software organizations, CMMI is more of a burden than anything else. GUnit, on the other hand, is a small and lightweight evaluation framework. So small, in fact, that when the cops ever search you or your vehicle, the chances of them busting you for it are pretty low. Of course they'll probably just plant a copy of GUnit on you anyway...

VI. Conclusion

Maintaining the street cred of a source code base has always been a high priority for software developers. However, up until this point there has been no framework for

automatically creating, running, and evaluating this particular metric. Therefore in order to fill this gap we have developed the GUnit testing harness. Source code is currently available online [4].

VII. ACKNOWLEDGMENTS

The author would like to thank you, dear reader, for making it this far. The metaphors never quite worked, but he kept with it the whole time, and that's got to count for something. The author would also like to thank the SIGBOVIK program commitee for graciously extending the submission deadline. If he had actually taken advantage of that extra time, this paper would no doubt be much funnier.

REFERENCES

[1] Website for the Eclipse SDK. `http://www.eclipse.org/`

[2] stic.man, M-1. Revolutionary but Gangsta. *Journal of the American Underground.* March 30, 2004. pp. 1045-1102.

[3] S. Radamenton, A. Robertson, J. J. Walker. *CMMI: Guidelines for Crunkability Improvement.* Addison-Wesley, 2003.

[4] GUnit Testing Harness Website. `http://www.g-unitsoldier.com/`

[5] W. C. Roeslisberger. z80 Assembly: Fad or In it to Win it? *4th Intl. Working Conference on 10x Programmers: Habits, Moods and Whims Track.* February 13, 1989.

[6] C. Smith. M. J. Blige. All That I Need. *Journal of the American Underground.* February 1, 1996.

[7] Wikipedia entry, "Hpnotiq." `http://en.wikipedia.org/wiki/Hpnotiq`

A Theft-Based Approach to 3d Object Acquisition

Ronit Slyper*
Carnegie Mellon University

James McCann†
Carnegie Mellon University

Method	Casing?	$	avg(Δ_s)	max(Δ_j)	$\mathbb{E}(\Delta_j)$	avg(Δ_j)	discount (fing.)	score
Snatch	no	0.45	10	3	0.5	0	5	83.8
Snatch	yes	0.45	5	3	0.5	0	5	77.5
Jack	no	15k	3	6	0.2	0	5	129.1
Jack	yes	15k	2	6	0.2	0	5	42.2
Abduct	no	-	122	12	7.5	1.2	4.5	71.9
Abduct	yes	-	64	12	7.5	0	5	66.3
Con	no	1G	40	9	1	0	5	97.2
Con	yes	1G	25	9	1	0	5	22.1

Figure 1: A comparison of various approaches to theft, with and without casing. Item value is given by $ (in dollars), time for theft is given in Δ_s (in cm-hours per foot). The empirically encountered jailtime is avg(Δ_j), while max(Δ_j) and $\mathbb{E}(\Delta_j)$ are calculated via lawyer; all are given in months.

Abstract

In graphics, one often wishes to acquire an accurate representation of an object. Much work has focused on novel devices for acquiring various properties of given objects, including geometry, surface reflectance, deformation modes, and even sound response. Devices have ranged from laser scanners to camera arrays and robotic probes, with scanning and processing times ranging from minutes to days. We present a novel technique that allows the acquisition of a complete representation of a wide range of 3d data sets in a few minutes (faster with hardware acceleration). The representation thus obtained can be photorealistically rendered at haptic rates – allowing a wide range of direct interaction and display possibilities.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation—Digitizing and Scanning; H.3 [Information Storage and Retrieval]: Information Search and Retrieval—Retrieval Models

Keywords: data acquisition, scanning, haptics, illicit activity

1 Introduction

In this paper we present a novel approach, borrowed from the criminal underworld, for the acquisition of complete 3d datasets. Datasets acquired with this approach are suitable for real-time rendering and haptic interaction.

*e-mail:rys@cs.cmu.edu
†e-mail:jmccann@cs.cmu.edu

Require: obj an object
if obj small **then**
  return Snatch(obj)
else if obj self-powered **then**
  return Jack(obj)
else if obj animate **then**
  return Abduct(obj)
else if obj controlled by person **then**
  return Con(obj)
else
  return $\emptyset$
end if

Figure 2: Description of the procedure Steal(obj). A simple set of heuristics is required to determine which method to use.

2 Related Work

As a complete object-data capture process, our technique stands to replace several conventional methods (at least in certain application domains). We provide a short review of these methods below.

Three-dimensional scanning is a popular technique for acquiring object geometry, with laser-based [Roach 1997] approaches already commercialized and widely used. Additional scanning systems rely on projectors or robotic probes [Bovik 1704]. Some scanners are additionally able to capture a primitive surface model.

For more complete surface model representation we must turn to appearance modeling and capture. Work in this field seeks to synthesize material BRDFs which match real materials, including skin [Hefner 1953; Silberstein 1965].

3 Method

3.1 Acquisition

In lieu of an elegant description, we present obtuse pseudocode with possibly undefined functions (Figure 2) – as has long been the tradition in computer science [Knuth 1981].

3.2 Rendering

One of the benefits of theft-based acquisition is that the data sets so obtained are *self-rendering*. That is, they are self-contained and able to selectively interact with photons in order to create a photo-realistic emission field. Such data sets also have *physical presence* which allows one to interact with them in a realistic and wholly satisfying manner.

3.3 Extensions

Like many approaches in graphics, our method may benefit from hardware acceleration. We find that a minimal set of hardware (e.g. pry-bar, lock-pick set, skeleton key, toothpaste) suits many possible applications, while more complicated systems (crane, flat-bed truck, submersible) may be required for more extreme situations. This systems allows a flexible cost-benefit trade-off.

Finally, we found that by using a pre-casing phase (so named because it involves "casing the joint" and "scoping things out") we can avoid common algorithmic pitfalls (security systems, alarms) and increase overall efficiency. It is also during this pre-casing phase that we are able to better optimize our selection of hardware accelerators.

4 Evaluation

In order to evaluate our methods, we tested our approach on several data sets. Theoretical bounds on jailtime (Δ_j) were calculated from criminal justice system records. Averages are presented over a number of trials sufficient to satisfy our material longings and kleptomania. In order to provide a reasonable comparison, we use the unqualified-apathetic-rand norm, as presented in Equation 1 (we set λ based on time of day).

$$\int_{\text{whatever}} \arccos \Delta_j - \frac{\$^2}{\lambda \cdot \Delta_s} \tag{1}$$

5 Conclusions

In this paper [Slyper and McCann 2007] we have presented a new approach to the acquisition of 3d objects. We hope, in the future, to become wealthy and well-taken care of by judicious application of this approach.

One drawback of our approach is the large Δ_j. While our heuristics do a reasonable job of avoiding a large Δ_j, this could be further mitigated with a proxy-based approach. In this approach, left for future work, one pays a third party (normally a member of the local "crime syndicate") to perform the acquisition.

Another method, which seems to warrant further investigation, is the direct purchase of goods. While it may seem counterintuitive, the authors have found that in some cases, costs can actually be lower than theft – and Δ_j is significantly reduced.

Acknowledgments

Thanks to Frankie (Pen. State) and to Big Joe (State Pen.), for key hints. Additional thanks to Fast-Fingered Earl for inspiration and methodological discussion. Condolences to the Stop-n-shop, East Plaza Mall, and Marty's Gas-n-Guns. Finally, thanks to our attorney Paws and his cat Edward for contributions during the evaluation phase. This research was supported by IRS grant 15-1040. Ronit Slyper was additionally supported by an NFL fellowship.

References

BOVIK, H. 1704. Robotic probes and you. *Probe-based Robotics 43*, 3, 105–205.

HEFNER, H. 1953. *Playboy*, vol. 1. Playboy Enterprises, Inc., December.

KNUTH, D. E. 1981. *Seminumerical Algorithms*, second ed., vol. 2 of *The Art of Computer Programming*. Addison-Wesley, Reading, Massachusetts, January.

ROACH, J., 1997. Austin powers: International man of mystery.

SILBERSTEIN, D. 1965. *Penthouse*, vol. 1. Penthouse Media Group, May.

SLYPER, R., AND MCCANN, J. 2007. A theft-based approach to 3d object acquisition. In *ACH SIGBOVIK*, vol. 1, 79–87.

Crash n' Compile: A formalization and empirical study of developer productivity and software quality through intoxication

Ciera Christopher

April 1, 2007

Abstract

The CrashNCompile process has become a topic of increasing interest in our community, though up to this point it has been discussed in an informal manner. In this research, we formalize the rules of the CrashNCompile process using an operational semantics. We verify the correctness and termination of these rules through observation and wavy-hand-proofs. In this study, we analyze the impact of the CrashNCompile process on developer productivity and software quality. We also analyze the many variants of the CrashNCompile process, including choice of language, compiler, and intoxicating beverage. We conclude by showing that "Team Distraction", a team comprised entirely of people with "fuzzy" majors, does indeed have an impact on the quality of the code produced, though it may in no way reflect the original requirements.

References

http://crash-n-compile.eorbit.net/index.html
http://www.langston.com/Fun_People/1995/1995AOA.html

Cycle Depletion – a Worldwide Crisis[1]

Joseph M. Newcomer[2] and Charles B. Weinstock[3]

In 1982, the authors did not publish a paper they wrote on the then-obvious Cycle Depletion problem (CDP). Twenty-five years later, we are not publishing a twenty-five-year retrospective paper on this problem.

Sadly, the cycle depletion problem continues to worsen. There are predictions that we face the imminent danger of a cycle depletion crisis by the year 2020. This paper should serve as a warning to everyone that we must address the cycle depletion problem immediately or face the consequences.

In 1982 we had observed that the cycle depletion problem was already serious.

The cycle depletion problem arises because there are a finite number of cycles in the Universe, and computers are depleting these at a ferocious rate.

Computers suck down cycles and emit heat and computation. Without cycles there can be no computation. This is the fundamental principle that makes all our computers run. The number of cycles sucked down by any computer has been increasing (although we will see that there have been changes that have reduced the need for cycles by six orders of magnitude, thus allowing us to be in what appears to be a steady-state situation).

A result we should mention here was with the launch of the Cycle Isotropy Observatory (CIO) in 1992, it became clear that cycles are anisotropically distributed throughout the Universe, with heavy concentrations in some places and far less dense concentrations elsewhere. The current best cosmological theory is that our solar system, and for light years around us, is a cycle-poor region to start with, and therefore we are already at a disadvantage. Work by Hawking [Hawking99] show that cycle densities near black holes are incredibly high. Mathematically, the number of cycles available at the event horizon is infinite, but we simply have no technology to tap such cycles.


Ptolemy
(Wikipedia)

Cycle theory is nothing new. The earliest recorded work dates back to Ptolemy (150 A.D.) who postulated that the Universe consisted of cycles and epicyles. We now know that epicycles are not required, and cycles are all there are. But, like Democritus and his theory of atoms (460 B.C.) he was millennia ahead of his time. However, it wasn't

[1] Portions of this work were unknowingly funded by a grant from Tartan Laboratories, Inc.
[2] FlounderCruft, Inc. Email: newcomer@flounder.com
[3] Software Engineering Institoot

until modern Quantum Cycle Theory evolved that we began to truly understand the fundamentals of cycles in the Universe. As this theory has matured, it has led to the Schwinn postulate of Quantum Computers, in which all problems are either solved or not solved, and until you read the solution, you don't know if the problem is solved. (There is some question as to whether or not a Quantum Computer can solve the Halting Problem, and there is at least some fear that if such a problem were presented to a Quantum Computer, the computer would implode to a singularity and suck down all the cycles for hundreds of light years around. Others assert that this is the explanation of the Fermi Paradox: any sufficiently advanced technological culture eventually gets to the point where they try this experiment, and are immediately reduced to using slide rules, rendering them incapable of solving the kinds of problems that would lead to interstellar communication or faster-than-light travel). Quantum Cycle Theory also allows us to explain Cycle Anisotropy (CA), although there are some that say there is no possible way to explain CA

The current value of the Critical Density Ω_Λ is such that the Universe appears to be permanently expanding, meaning there is only one cycle (a so-called uni-cycle) to the Universe (there will be no Big Crunch followed by another Big Bang), so the dominant single metacycle means that all available cycles are merely subdivisions of this one cycle. Ultimately, a cycle may not be the atomic unit of computation, but the equivalent of quarks has not yet been determined. Nonetheless, there seems to be only one original cycle from which all others derive. This is one of the open questions of Quantum Cycle Theory.

A competing theory, derided by some as not being a theory at all, is that the one distinguished cycle is the source of all other cycles. Even within this theory there are competing points of view with some believing in the so-called Unicycle, and others in the so-called Tricycle – the latter having more adherents in the community. Regardless, both communities believe that the distinguished cycle designed and created all of the other cycles.

A theory of Vacuum Cycles suggests that cycles may spontaneously be created in the vacuum of space; however, they will be created with a corresponding anticycle and the two will cancel out according to the formula $e = MC^2$, but given that cycles may be massless, this means no energy would be produced. There is no good explanation of why our Universe favors cycles over anticycles.

We first observed the cycle depletion problem at CMU in 1976, when we moved our 16-processor multiprocessor, C.mmp, into the main computer room with our KL-10 processor. Shortly thereafter, the KL-10 began to experience various problems in reliability.

This led to one of those engineering-vs.-science debates. The engineers, lacking any solid theoretical basis, asserted that the cause was that the 16-processor system generated too much heat and overloaded the air conditioning system, raising the temperatures and causing failures. The scientists, on the other hand, with a firm grasp of cycle theory

(even in its early form in those days), knew that the real problem was that the 16-processor system caused a local depletion in the time-space-cycle continuum, sucking cycles away from the single-processor KL-10 and causing it to fail.

In those heady days, it was believed the number of available cycles were unlimited, or at the very least good for centuries. There were few efforts to conserve cycles.

Early IBM mainframes were equipped with a "usage clock" that ran when the computer was computing. This measured the amount of usage of the computer, and hence the monthly rental cost. [We're not making this up]. If the CPU issued a HALT or WAIT instruction, the CPU clock stopped running. While advertised as a way of reducing costs, it was clear that IBM was not creating cycles, and therefore (had cycle theory been known) was apparently charging for the use of a natural resource. But the opposing theory is that researchers at T.J. Watson Research Center had already discovered cycle theory, had grossly underestimated the available number of cycles (they had once been said to have stated that the time-space-cycle continuum could support no more than 12 computers), and had actually done this as a way of forcing their end users to conserve cycles by disguising it as saving money.

Those of us who "grew up" in the 1960s had an awareness of conservation that seems to have disappeared in the 1980s through the present. Many of us would work late into the night, sometimes overnight, to ensure that no cycles were wasted. Our more cynical colleagues said that we were doing this because we got better response time, but the real reason was that we could not bear to see cycles being wasted.

[Jack McCredie, faculty at the Carnegie Mellon University Computer Science Department at that time, observed wistfully at one CS party one evening in 1972, "Think of all those cycles going over the dam". This was typical of the concerns we had at that time].

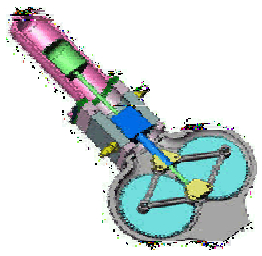
Harry Bovik did one of the fundamental experiments in disproving the *feng shui* theory of computing. This theory stated that the proper alignment of a computer with the cycle ether would result in more reliable behavior. In the Michaelson-Bovik experiment (unpublished), a computer was placed on a platform and operated for a period of time. The platform was then rotated 90º and operated for the same period of time. No change in the speed of computing or the reliability of the computer was measured (within experimental error), thus disproving the idea that cycles exist in the cycliferous ether.

Moore's Law has saved us. As computers are built with smaller and smaller design rules, the actual number of cycles required has gone down as the same rate that the cycle speed has increased. A microcomputer uses microcycles (10^{-6} cycles), so a million computers executing microcycles consume about the same amount as a old mainframe executing cycles. Intel architectures, for example, do not execute instructions (in the CISC sense) but instead compile those instructions into micro-ops (μ-ops), and consequently we have been largely oblivious to the magnitude of the problem.

Federal standards for cycle conservation have not been well-enforced, particularly under the Bush administration, which fired all eight of the Federal Cycle Inspectors that had been hired by the Clinton administration and signed a contract with Dubai-based Hallibutron. Modern chips do have some of the cycle-saving mechanisms mandated by the Cycle Conservation Act of 1994 (a law largely spearheaded by Al Gore). These include mechanisms to slow the clock down under conditions of low usage, thus reducing the absolute cycle requirements; having sleep states and hibernation states that reduce cycle usage considerably. A public that is hungry for cycles would not accept these limitations for the sake of simple conservation (witness our treatment of the oil shortage issue), so these cycle-saving techniques are marketed as mechanisms for saving power (economic incentive) or extending battery life (self-interest incentive), but the truth is that they are there to attempt to reduce cycle usage.

Computer vendors continue to be afraid to let the truth be known about the upcoming cycle crisis. We already have evidence that IBM in the 1950s knew about this problem, and any vendor directly asked about this issue will deny it is a problem, while behind the scenes they work hard to deal with cycle conservation.

The increase in graphics requirements to support Symbolic User Virtualization, such as is used in video games, is another source of cycle depletion; graphics cards are simply very sophisticated domain-specific computers. There is serious concern about the impact of SUV popularity on our limited resources.


Stirling Engine (Wikipedia)

Since cycles are converted to heat, there are some thoughts that we should be able to have some way to convert heat to cycles. Unfortunately, the only effective state-of-the-art device we have is the Stirling Engine, which was invented in 1816. No more effective mechanism for converting heat to cycles has been devised. In principle, this engine can work at 80% of the theoretical Carnot efficiency, but it just doesn't produce enough cycles to make a difference. However, advances in nanotechnology could change this. Currently, nanotechnologists working on various kinds of mechanical effectors have not been able to understand the need for micro-Stirling engines.

Back in the 1960s, far-sighted pioneers such as J.C.R. Licklider saw the coming catastrophe, and created the ARPANet. One of the stated purposes of the ARPANet was to allow cycle-sharing, specifically, to allow users anywhere to share the computing on some non-colocated processor. This would allow them to place large computing facilities at the sites of major cycle concentrations, but users in cycle-poor areas could use these cycles.

NSF has funded a few supercomputer centers. Their apparent random distribution is not random at all; they are all located at sites of massive cycle concentrations.

Back when ILLIAC-IV was first installed, the original plan had been to install it at the University of Illinois campus. As an almost last-minute decision, the computer was located at the NASA Ames research facility at Moffet Field in California. The public reason given was that this was done because they were afraid that student antiwar activists might break into the computer room at the University and damage or destroy the computer. This story was simply for public consumption. Careful studies had shown that U. Ill. was in a cycle-poor region (no one who had chosen the original site was aware of cycle theory) whereas NASA Ames had been built (deliberately) in an area of high cycle availability.

Not all computer sites have been so fortunate. The Livermore Laboratories Supercomputer Center had been located at a cycle-rich site, but in later years, for reasons not yet fully understood (although Quantum Cycle Theory is beginning to yield some results) the nexus of cycles moved. This required either moving all of Livermore, or somehow creating more cycles, so a cyclotron was built there. Unfortunately, these are not cost-effective for everyday computing, costing tens of millions of dollars to construct and requiring military-grade budgets to keep running.

Not many people realize the true purpose of the World Wide Web. CERN suffered from the same problem as Livermore, and in their research to solve the cycle problem, the WWW emerged. Although the public cover story is that it makes information readily available, the real truth is that it is a network for importing cycles from cycle-rich countries to cycle-poor countries. The reason many emerging countries are spending such massive efforts on their network infrastructures is that they hope to become major cycle exporters in the next decade.

Not all emerging economies are in this state, however. The massive adoption of personal computing in India and China will soon mean that these countries will become major cycle importers as well. Neither of these countries are signatories to the Sapporo Accords (named after the beer drunk at the sushi restaurant where these accords were proposed), and the U.S. has stated that as long as these countries are not signatories, there is little that the U.S. can do internally to reduce its cycle consumption.

The real risk as that the cycle consumption in these countries will only exacerbate the growing cycle crisis, and competition for the remaining cycles from small but cycle-rich countries will result in serious international tensions. Estimates are that wars over cycles will be inevitable by 2030.

What can we do? We must reduce our cycle consumption. Estimates of 10% per year improvement will give us time to find alternative cycle sources, and there are promising research directions that may yield solutions in the future. Meanwhile, the use of slide rules and abacuses (abaci?) should be encouraged. PDAs should be replaced by calendar notebooks. Laptops should be reserved for holding cats, not computers. Besides, purring cats on one's laptop are far more soothing than computers, anyway.

References

[Hawking99] Hawking, Stephen, On the Diminishing Computer Reliability in the Presence of Universal Anamolies, *The Journal of Isotrophic Physics*, Vol. 17, No. 7, July 1999, pp1346-1378.

Track III:

The Meta-Art of Paper-Writing

Abstracter Abstracts*

Daniel K. Lee

Carnegie Mellon University

Abstract

We did it!

*This work is partially supported by the National Science Foundation under a Graduate Research Fellowship and D's Six Pax and Dogz.

Qualitative Methods for Interagent Communication

Jason Reed*
jcreed@cs.cmu.edu
School of Computer Science
Carnegie Mellon University

Abstract

This one time I chatted up a girl at a party, and it ended in disaster. Don't believe anything she says about it.

Keywords: Anthropomology, Relational Algebra

1 Introduction

Hey there! Hi. How's it going. Peggy? Hi, Peggy. Nice to meet you. You know, my older sister's [SA83] name is Peggy, too. Yeah, really! No, I don't think so. Hah, hah! So you came with — ? How'd you meet him? Yeah? So you two are — you know? Oh? No? Oh, okay, I didn't mean to suggest — heh!

2 Background

So you're Irish? Yeah, the hair is a dead giveaway, isn't it. Me, well, on my mom's side, I'm French Canadian, wood pulp, rag stock, and laser printer toner, and on my dad's side, it's a mix of LaTeX, ASCII, and Estonian. You wouldn't believe the teasing I got as a kid!

Boy, the stories I could tell you about — oh? You have to go to the bathroom? Okay. I'll still be here when you get back.

3 Related Work

Hey again! Anyway, this one time my little sister [Ali85] got submitted to a pretty major journal, and they turned her down just because — and I'm told this on good authority — she had a few mistakes involving substitution 'its' for 'it's'. The injustice! Who doesn't have a few blemishes at that stage growing up, really? It's a difficult time. Three rejections later, she finally gave up and sent herself out to a workshop on the Cybernetics of Peer Review Science. My dad [Ali51] still hasn't forgiven her. He thinks it's just an internet scam — I admit the fees were suspiciously large — but it's done wonders for her self-esteem.

4 Development

Boy, once i * get a few of those PSPACE-hard lemonades in me i just dont know hwat im saying anymore, you know? you know? i guess i really dont have my i really dont have much tolera i mean tolerance for that kinda thing heeee! i guess you don't either. i don't know where that bucket came from either, but you sure tripped over it good Here, let me help you ups. * Up! Come up! there you go! Anyway i was wondering if you wanted to come back to my place and uh **hic** *overfill my hboxes* if you drift my catch

5 Details of Private Interaction Protocol

Deferred to the Technical Report, to be released June 2007 on Citeseer Adult Video and Academic Publishing.

*The author would like to thank this paper for writing itself.

6 Open Problems

Hey! You're not supposed to look in there! That's where I keep my old review forms! Those are supposed to be private! No! No! Don't read them! Stop it! Stop reading them! Stop laughing! Stop! Please! No! Why are you leaving? Give that back! Pleeeease! Please don't tell anyone he said I'm 'conceived poorly, articulated worse, and quite possibly the result of plagiarism so incompetent as to resemble a peculiar originality, but one inept far beyond the capabilities of the average researcher'! Come back!

References

[Ali51] Inter Alia. Denotational higher-order harrop heredity. *J. Hoity Toitology*, 1(1):1–137, October 1951.

[Ali85] E. T. Alia. Hyper-driven device drivers. Draft Manuscript, August 1985.

[SA83] Verba Sesquiped-Alia. Peg-passing procedural protocols. In R. E. D'actor, editor, *Programming with Peg Procedures (PPP'83)*, pages 221–230, Käseburg, Wisconsin, February 1983. Verlag & Sons Publishing Co.

The Epistemology of Exploratory Empiricism: Science

Maoz E. Berlinger, Tom E. Helostpants, Valiant E. Jeenen, Flossyfry E. Jest, and Sen E. Rajdoc

Departments of Artifactual Engineering and Hermeneutical Computing, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh PA 15213-3890, USA

Abstract

In this paper we explore a sequential linear incremental methodology and its applications to epistemelogical questions related to exploratory empiricism. By solving the commencement problem we elucidate the complex relationship between Barthmodal grammars and scientific progress, and demonstrate that solving epistemelogical questions is equivalent to scientific discovery. We also briefly discuss bananas.

1 Introduction

In this paper we consider the intersection of that which we believe based on evidentiary empiricism extended to novel areas (exploratory empiricism) with that which is actually true; in other words, we are seeking to appropriately categorize information gained through exploratory empiricism as both trustworthy and credible or as failing on one or both of those counts.

The principal criterion we shall employ as a barometer in estimating both of the above categorical indices (i.e., trustworthiness, credibility) is derived from approximating the magnitude and etymological/lexical entropy of sentential units generated by an agent formulating exploratory-empirical hypotheses about the world, wholly divorced from the identity of that agent, and in fact from the semantic content of the aforementioned sentential units, thus ensuring a strong form of objectivity and impartiality.

Our method is at once absolutely *necessary* while also *utterly* paradoxical. It can be traced back to Archimedean philosophy, which embodied both evidentiary-based methodologies for knowledge acquisition and metaphysical contemplations on the validity of abstract knowledge expressed in a socially constructed format, i.e. language, and therefore raises key questions about the role of linguistics in empiricism in general, and exploratory empiricism specifically.

2 Bananas and Barth-modal T-grammars

Consider the case of bananas. It is made of sturdy wood, and holds two gross of this yellow, succulent fruit. Science has determined that the Cavendish banana is likely to become extinct if certain fungi (considered by the present author to be highly symbolic of the several threats today facing the scientific establishment) have their way. Perhaps you should eat a few bananas (see Figure 1) right now, before reading another paragraph, and consider the fleeting opportunity you have, to wit, to accept this paper and its contribution.

To ourselves we ask, "Why don't you write something already?" The hermeneutical implications of this question weigh heavily on our minds, and in order to avoid answering them we choose to write about them. No doubt the recent discovery locating evidentiary empiricism in the exploratory domain within the category of nonmodal Barth-modal T-grammars makes the "commencement problem" formidable; fortunately, our identification of the Snaxx combinator for this T-grammar ("Dubuque, Iowa"; cf. Appendix) mitigates the challenge somewhat.

Fig. 1. Yellow, succulent.

Previous reviewers have challenged our use of place-names as citations, but we pose the following question: Who *has* been to Dubuque, Iowa? Really? Perhaps the streets there are littered with Barth-modal T-grammar Snaxx combinators, for all you know.

As is evidenced by the reader having progressed thus far in the paper, we have produced a satisfactory solution to the commencement problem via an intermittent constructivist approach, which paradoxically relies on social deconstruction efforts focused on global sales of bananas.

We have used our technique to analyze itself - self-referentially applying our linguistic analyses to approximate the trustworthiness of this very paper - and in doing so revealed unprecedentedly high levels of credibility in ourselves. This confirms our (now empirically credible) belief that by combining the epistemologically sound and theoretically grounded elements of Archimedean philosophy with the practical lessons of evidentiary empiricism (re: banana appreciation), we have isolated for the first time in its pure form the true essence of Science itself.

3 Conclusions

We're going to win best paper award! We're going to win best paper award! We're going to win best paper award!

4 Acknowledgments

We gratefully acknowledge support from the Office of Dense and Opaque Research (ODOR) grant number DNS-028841971 and the Dole Food Company grant number BAN-693997510.

APPENDIX

"Dubuque, Iowa"

Synchronistic Hyperparadigmatism

Connor Sites-Bowen

March 20, 2007

Abstract

M. Deschamps was in his teens served a plum pudding by one M. Fortgibu. In his thirties, M. Deschamps ordered plum pudding at a restaurant, only to be informed that a M. Fortgibu had just been served the last piece. Thirty years later, M. Deschamps was treated to plum pudding at a highly select party. He explained the earlier plum pudding incidents and remarked that now only M. Fortgibu's presence was needed to truly complete the scene. Just at that instant, a delirious and senile Fortgibu entered the party, having mistaken it's address for that of the engagement he was supposed to be attending that night. This is Carl Jung's most famous example of synchronicity (or serendipity).

When caught in the throws of a serendipitous flow of time, one often wonders if all such experiences have similar phenomenological characteristics; Does synchronicity have a higher structure? Can it be invoked, or if already flowing can it be ridden to a more complete conclusion? Can a run of serendipitous events be revived after it has seemingly lapsed? Does meta-synchronicity exist? This paper examines multiple accounts of synchronicity, as well as other acausal time-space events, in an effort to find a meta-form for Acausal Parallelism, which can also function as a hyper-paradigm for the general flow of time. Such a hyper-paradigm is then presented as a useful philosophical tool to analyze ESP, travel through time, direct manipulation of synchronistic events, magic, alien abduction, and other parapsychological processes, especially when superimposed onto a seemingly causal universe.

A Systematic Evaluation of the Observed Degradation of Typesetting Technology in the 20th Century

Reginald J. Qnuth

March 3, 2007

Abstract

We systematically evaluate typesetting technology over the course of the 20th century and discover an astonishing degradation. We hypothesize on the potential causes of this observed degradation and conclude that it is the work of malicious time-travelling monkeys.

Keywords: type, systems

1 Introduction

In this work, we advance the hypothesis that typesetting technology took a dangerously steep downward turn in the latter half of the 20th century. This is not a new hypothesis; the world-renowned computer science genius Knuth made similar observations in the thick of the matter 30 years ago in March 1977, saying of his gloriously comprehensive compendium *The Art of Computer Programming* [2], "I had spent 15 years writing those books, but if they were going to look awful, I didn't want to write any more" [3].

We take Knuth's hypothesis and validate it with a systematic and unbiased evaluation of over three academic papers selected at random from between the years 1900 and 1999. In Section 1, we introduce our hypothesis. Section 2 discusses our raw data in detail. Finally, in Section 3 we draw the startling conclusion that typesetting technology actually degraded over the course of the 20th century!

2 Experimental data

2.1 1936: the heady days of the decision problem

Hilbert's 23 problems began a century of glory in mathematics. In 1936, Alonzo Church published a note [1] on what is widely regarded as "by far the coolest of Hilbert's 23" [8], the Entscheidungsproblem. We excerpt this note in Figure 2.

(Church is also well-known and highly regarded for his work on the hat calculus; see [4] for a contemporary tutorial introduction.)

This publication represents the pinnacle of publishing quality in the 20th century, with its fully-typeset mathematics and proportionally spaced fonts. No characters are hand-drawn, and the kerning is superb. The footnotes aren't even fragile! An exemplary exemplar of style and quality—precisely what we've come to expect from an academician as talented as Church.

Figure 1: Alonzo Church, a very talented academician. (Cheerful, too!)

THE JOURNAL OF SYMBOLIC LOGIC
Volume 1, Number 1, March 1936

A NOTE ON THE ENTSCHEIDUNGSPROBLEM

ALONZO CHURCH

In a recent paper[1] the author has proposed a definition of the commonly used term "effectively calculable" and has shown on the basis of this definition that the general case of the Entscheidungsproblem is unsolvable in any system of symbolic logic which is adequate to a certain portion of arithmetic and is ω-consistent. The purpose of the present note is to outline an extension of this result to the engere Funktionenkalkül of Hilbert and Ackermann.[2]

In the author's cited paper it is pointed out that there can be associated recursively with every well-formed formula[3] a recursive enumeration of the formulas into which it is convertible.[3] This means the existence of a recursively defined function a of two positive integers such that, if y is the Gödel representation of a well-formed formula $\boldsymbol{Y}$ then $a(x, y)$ is the Gödel representation of the xth formula in the enumeration of the formulas into which $\boldsymbol{Y}$ is convertible.

Consider the system L of symbolic logic which arises from the engere Funktionenkalkül by adding to it: as additional undefined symbols, a symbol 1 for the number 1 (regarded as an individual), a symbol $=$ for the propositional function $=$ (equality of individuals), a symbol s for the arithmetic function $x+1$, a symbol a for the arithmetic function a described in the preceding paragraph, and symbols $b_1, b_2, \cdots, b_k$ for the auxiliary arithmetic functions which are employed in the recursive definition of a; and as additional axioms, the recursion equations for the functions $a, b_1, b_2, \cdots, b_k$ (expressed with free individual variables, the class of individuals being taken as identical with the class of positive integers), and two axioms of equality, $x=x$, and $x=y \rightarrow [F(x) \rightarrow F(y)]$.

The consistency of the system L follows by the methods of existing proofs.[4] The ω-consistency of L is a matter of more difficulty, but for our present purpose the following weaker property of L is sufficient: *if* P *contains no quantifiers and* (Ex)P *is provable in* L *then not all of* $\overline{P}_1, \overline{P}_2, \overline{P}_3, \cdots$ *are provable in* L (*where* $P_1, P_2, P_3, \cdots$ *are respectively the results of substituting* 1, 2, 3, $\cdots$ *for* x *throughout* P). This property has been proved by Paul Bernays[5] for any one of a class of systems of which L is one. Hence, by the argument of the author's cited paper, follows:

Received April 15, 1936.

[1] *An unsolvable problem of elementary number theory*, ***American journal of mathematics,*** vol. 58 (1936).

[2] ***Grundzüge der theoretischen Logik,*** Berlin 1928.

[3] Definitions of the terms *well-formed formula* and *convertible* are given in the cited paper.

[4] Cf. Wilhelm Ackermann, *Begründung des "tertium non datur" mittels der Hilbertschen Theorie der Widerspruchsfreiheit,* ***Mathematische Annalen,*** vol. 93 (1924–5), pp. 1–136; J. v. Neumann, *Zur Hilbertschen Beweistheorie,* ***Mathematische Zeitschrift,*** vol. 26 (1927), pp. 1–46; Jacques Herbrand, *Sur la non-contradiction de l'arithmétique,* ***Journal für die reine und angewandte Mathematik,*** vol. 166 (1931–2), pp. 1–8.

[5] In lectures at Princeton, N. J., 1936. The methods employed are those of existing consistency proofs.

Figure 2: 1936 publication.

2.2 1974: a less innocent age

Flash forward to the year 1974. Nixon faces impeachment for the Watergate scandal. India successfully detonates its first nuclear weapon. Polymorphism is in its fledgling stages.

Enter John Reynolds.

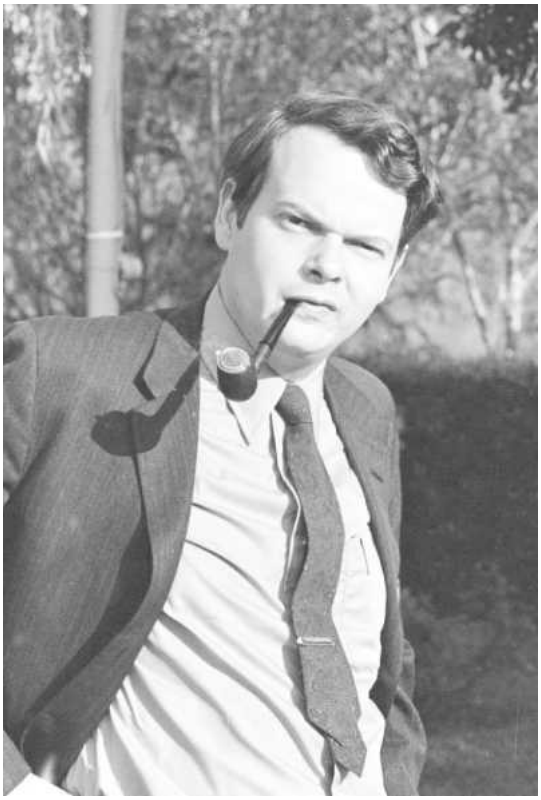


Figure 3: Bright-eyed and bushy-tailed John Reynolds, with a pipe.

It was in this year that Reynolds published his monumental manuscript on the polymorphic λ-calculus [5]. Despite being an academic work of the highest quality [XX cites???], its typesetting left much to be desired.

As one can see clearly from the scan in Figure 4, 1974 marked an age of "digital typography"—characters unavailable on the standard typewriter were drawn in by hand. Examples include the characters $\forall$, $\subseteq$, $\bigsqcup$, and (particularly damningly) $\mathcal{D}$. The careful reader will note the scribbly nature of the tail on the $\mapsto$ arrow. But at least we have a full complement of Greek letters, and everything is sufficiently well-spaced to be legible ...

The definition of the functor delta is less obvious. For all functors θ from C to C, delta(θ) is the complete lattice with elements

$$\{ f \mid f \in \prod_{D \in \mathcal{D}} \theta(D) \text{ and } (\forall D, D' \in \mathcal{D})(\forall \rho \in rep(D,D'))\theta(\rho): f(D) \mapsto f(D') \}$$

with the partial ordering $f \sqsubseteq g$ iff $(\forall D \in \mathcal{D})\ f(D) \sqsubseteq_{\theta(D)} g(D)$. For all natural transformations η from θ to θ',

$$delta(\eta) = \langle \lambda f \in delta(\theta).\ \lambda D \in \mathcal{D}.\ [\eta(D)]_1(f(D)),\ \lambda f \in delta(\theta').\ \lambda D \in \mathcal{D}.\ [\eta(D)]_2(f(D)) \rangle .$$

At this point, we must admit a serious lacuna in our chain of argument. Although delta(θ) is a complete lattice (with $(\bigsqcup F)(D) = \bigsqcup_{\theta(D)} \{f(D) \mid f \in F\}$), it is not known to be a domain, i.e., the question of whether it is continuous and countably based has not been resolved. Nevertheless there is reasonable hope of evading the set-theoretic paradoxes. Even though $\prod_{D \in \mathcal{D}} \theta(D)$ is immense (since $\mathcal{D}$ is a class), the stringent restrictions on membership in delta(θ) seem to make its size tractable. For example, if $f \in delta(\theta)$, then the value of f(D) determines its value for any domain isomorphic to D.

The definition of delta and the properties of representations give the lemma:

Let η be a natural transformation from θ to θ', $f \in delta(\theta)$ and $f' \in delta(\theta')$. Then

$$delta(\eta): f \mapsto f'$$

if and only if, for all D, D' $\in \mathcal{D}$, and $\rho \in rep(D, D')$,

$$\eta(D') \cdot \theta(\rho): f(D) \mapsto f'(D') .$$

which, with the definition of B, gives:

Let $t \in T$, $w \in W$, $\bar{\rho} \in rep(\bar{D}, \bar{D}')$, $f \in B[\Delta t.\ w](\bar{D})$, and $f' \in B[\Delta t.\ w](\bar{D}')$. Then

$$B[\Delta t.\ w](\bar{\rho}): f \mapsto f'$$

if and only if, for all D, D' $\in \mathcal{D}$, and $\rho \in rep(D, D')$,

$$B[w][\bar{\rho}|t|\rho]: f(D) \mapsto f'(D') .$$

From the final lemmas obtained about arrow and delta, the representation theorem can be proved by structural induction on r.

Figure 4: 1974 publication.

2.3 1983: oh the burning!

Advance the clock 9 years to 1983. Ronnie Reagan leads the United States in their epic battle with the Evil Empire; the Star Wars project[1] plays a critical role. Meanwhile Luke and Leia lead the rebellion in *Return of the Jedi*. Reynolds does his part by proving the glorious Abstraction Theorem [6]. Reynolds has

Figure 5: Older, wiser John Reynolds.

grown older, and perhaps wiser, but the quality of his typesetting has diminished dramatically.

Figure 6 demonstrates the terrible turn taken by typesetting technology in this war-torn era. Although typewriting technology has acquired certain key characters ($\forall$, for example), this is only at the expense of the all-important capital Π, which is more hand-drawn, larger, and uglier than it's ever been in the history of life on earth [citation needed]! Agghhh! The math, it burns my eyes!

[1] More properly referred to as the Strategic Defense Initiative

where $s \to s'$ denotes the set of all functions from s to s' and $s \times s'$ denotes the set of pairs $\langle x, x' \rangle$ such that $x \in s$ and $x' \in s'$. Then the set $S^{\#}\omega$ is the meaning of the type expression ω under the set assignment S. Note that, when $\omega \in \Omega_c$, $S^{\#}\omega$ is independent of the set assignment S.

We use $*$ to denote a further extension from type expressions to type assignments: If S is a set assignment then $S^{\#*}$ is the mapping from Ω^* to the class of sets such that

$$S^{\#*}\pi = \prod_{v \in \mathrm{dom}\ \pi} S^{\#}(\pi v) \ . \qquad (S^*)$$

Then $S^{\#*}\pi$ is the set of environments appropriate to π and S.

A conventional semantics for ordinary expressions would be obtained by fixing some set assignment S and defining a family of semantic functions from $E_{\pi\omega}$ to $S^{\#*}\pi \to S^{\#}\omega$. However, to capture abstraction properties we will need to relate the meanings of an expression under different set assignments. For this reason, we will treat set assignments as explicit parameters of the semantic functions. Specifically, for each $\pi \in \Omega^*$ and $\omega \in \Omega$ we will define a semantic function

$$\mu_{\pi\omega} \in E_{\pi\omega} \to \prod_{S \in \mathcal{S}} (S^{\#*}\pi \to S^{\#}\omega) \ ,$$

where $\mathcal{S}$ denotes the class of all set assignments.

We assume we are given, for each $\omega \in \Omega_c$, a function

$$\alpha_\omega \in K_\omega \to S^{\#}\omega$$

providing meanings (independent of S) to the ordinary constants of type ω. Then the semantic functions are defined by

If $k \in K_\omega$ then $\mu_{\pi\omega}[\![k]\!]\ S\ \eta = \alpha_\omega\ k$, (Ma)

If $v \in \mathrm{dom}\ \pi$ then $\mu_{\pi,\pi v}[\![v]\!]\ S\ \eta = \eta\ v$, (Mb)

If $e_1 \in E_{\pi,\omega\to\omega'}$ and $e_2 \in E_{\pi\omega}$ then
$\mu_{\pi\omega'}[\![e_1(e_2)]\!]\ S\ \eta =$ (Mc)
$\mu_{\pi,\omega\to\omega'}[\![e_1]\!]\ S\ \eta\ (\mu_{\pi\omega}[\![e_2]\!]\ S\ \eta)$,

If $e \in E_{[\pi|v:\omega],\omega'}$ then
$\mu_{\pi,\omega\to\omega'}[\![\lambda v{:}\omega.\ e]\!]\ S\ \eta = f$ (Md)
where $f \in S^{\#}\omega \to S^{\#}\omega'$ is such that
$f\ x = \mu_{[\pi|v:\omega],\omega'}[\![e]\!]\ S\ [\eta|v:x]$,

If $e \in E_{\pi\omega}$ and $e' \in E_{\pi\omega'}$ then
$\mu_{\pi,\omega\times\omega'}[\![\langle e, e'\rangle]\!]\ S\ \eta =$ (Me)
$\langle \mu_{\pi\omega}[\![e]\!]\ S\ \eta,\ \mu_{\pi\omega'}[\![e']\!]\ S\ \eta \rangle$,

If $e \in E_{\pi,\omega\times\omega'}$ then
$\mu_{\pi\omega}[\![e.1]\!]\ S\ \eta = [\mu_{\pi,\omega\times\omega'}[\![e]\!]\ S\ \eta]_1$ (Mf)
$\mu_{\pi\omega'}[\![e.2]\!]\ S\ \eta = [\mu_{\pi,\omega\times\omega'}[\![e]\!]\ S\ \eta]_2$,

If $b \in E_{\pi,\mathrm{Bool}}$ and $e, e' \in E_{\pi\omega}$ then
$\mu_{\pi\omega}[\![\underline{\mathrm{if}}\ b\ \underline{\mathrm{then}}\ e\ \underline{\mathrm{else}}\ e']\!]\ S\ \eta =$ (Mg)
$\underline{\mathrm{if}}\ \mu_{\pi,\mathrm{Bool}}[\![b]\!]\ S\ \eta = \underline{\mathrm{true}}$
$\underline{\mathrm{then}}\ \mu_{\pi\omega}[\![e]\!]\ S\ \eta\ \underline{\mathrm{else}}\ \mu_{\pi\omega}[\![e']\!]\ S\ \eta$.

3. THE ABSTRACTION THEOREM

We now want to formulate an abstraction theorem that connects the meanings of an ordinary expression under different set assignments. The underlying idea is that the meanings of an expression in "related" environments will be "related" values. But here "related" must denote a different relation for each type expression and type assignment. Moreover, while the relation for each type variable is arbitrary, the relations for compound type expressions and type assignments must be induced in a specified way. In other words, we must specify how an assignment of relations to type variables is extended to type expressions and type assignments.

This can be formalized by defining a "relation semantics" for type expressions that parallels their set-theoretic semantics. For sets s_1 and s_2, we introduce the set

$$\mathrm{Rel}(s_1, s_2) = \{r \mid r \subseteq s_1 \times s_2\}$$

of binary relations between s_1 and s_2, and we write

$$\mathrm{I}(s) = \{\langle x, x\rangle \mid x \in s\} \in \mathrm{Rel}(s, s)$$

for the identity relation on a set s. For $r \in \mathrm{Rel}(s_1, s_2)$ and $r' \in \mathrm{Rel}(s_1', s_2')$, we write $r \to r'$ for the relation in $\mathrm{Rel}(s_1 \to s_1', s_2 \to s_2')$ such that

$\langle f_1, f_2\rangle \in r \to r'$ iff
$(\forall \langle x_1, x_2\rangle \in r)\ \langle f_1\ x_1, f_2\ x_2\rangle \in r'$,

and $r \times r'$ for the relation in $\mathrm{Rel}(s_1 \times s_1', s_2 \times s_2')$ such that

$\langle\langle x_1, x_1'\rangle, \langle x_2, x_2'\rangle\rangle \in r \times r'$ iff
$\langle x_1, x_2\rangle \in r$ and $\langle x_1', x_2'\rangle \in r'$.

In other words, functions are related if they map related arguments into related results, and pairs are related if their corresponding components are related.

For set assignments S_1 and S_2, a member of

$$\prod_{\tau \in T} \mathrm{Rel}(S_1\tau, S_2\tau)$$

is called a (binary) relation assignment between S_1 and S_2. Having defined $\to$ and $\times$ for relations we can extend relation assignments from T to Ω and Ω^* in essentially the same way as we extended set assignments. If R is a relation assignment between S_1 and S_2 then

$$R^{\#} \in \prod_{\omega \in \Omega} \mathrm{Rel}(S_1^{\#}\omega, S_2^{\#}\omega)$$

is such that

If $\kappa \in C$ then $R^{\#}\kappa = \mathrm{I}(CS\ \kappa)$, (R1)

If $\tau \in T$ then $R^{\#}\tau = R\ \tau$, (R2)

If $\omega, \omega' \in \Omega$ then
$R^{\#}(\omega \to \omega') = R^{\#}\omega \to R^{\#}\omega'$, (R3)

If $\omega, \omega' \in \Omega$ then
$R^{\#}(\omega \times \omega') = R^{\#}\omega \times R^{\#}\omega'$, (R4)

Figure 6: 1983 publication.

2.4 1990: hell on earth

Six years later, the Soviet Union collapses, and Ringard publishes his seminal preprint on mustard watches [7], shown in Figure 8.

Figure 7: Yann-Joachim Ringard—no relation to Jean-Yves "mad dog" Girard.

Although this superficially *seems* to represent an increase in publishing quality, our unbiased opinion is that this apparent increase is merely illusory. Figures are now entirely hand-drawn, and "smart quotes" are conspicuously absent. "QED" replaces the traditional □. Mathematical quality has taken a similarly downward turn; Ringard's so-called "proofs" can barely be called sketches.

1. Mustard watches : definitions and main results

Unless otherwise stated we use the term "watch" as an abbreviation for "analog pocket watch".

DEFINITION 1

Let W be a classical watch ; a *mustard watch* derived from W is any W' obtained from W by adding a certain amount of mustard in the mechanism. (see fig. 1)

It is immediate from the definition that a given classical watch may have several extensions into a mustard watch, whereas given a mustard watch W' there is only one underlying classical watch W from which W' is derived.

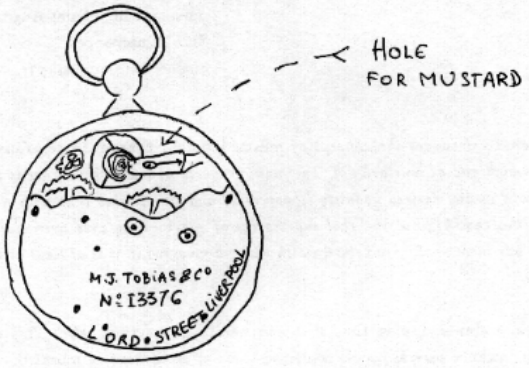


figure 1 : mustard watch just before refill

DEFINITION 2 :

A mustard watch is said to be *proper* when the amount of mustard in it is non zero ; it is said to be *degenerated* otherwise.

THEOREM 1 :

It is possible to get "as much mustard as wanted" from a mustard watch. More precisely, given any amount m of mustard, there is a mustard watch $W(m)$ containing mustard in quantity m.

PROOF : first observe that there are classical watches in any size ; in particular let W_* be one with enough room in the mechanism to harbour m grams of mustard. Consider the *completion* $W(m)$ of W achieved by adding m grams of mustard to W. QED

mustard watches

Figure 8: 1990 publication.

2.5 A graph

Any good experimental systems paper needs a graph. The graph in Figure 9 shows undeniably that typesetting quality has decreased between 1900 and 1999.

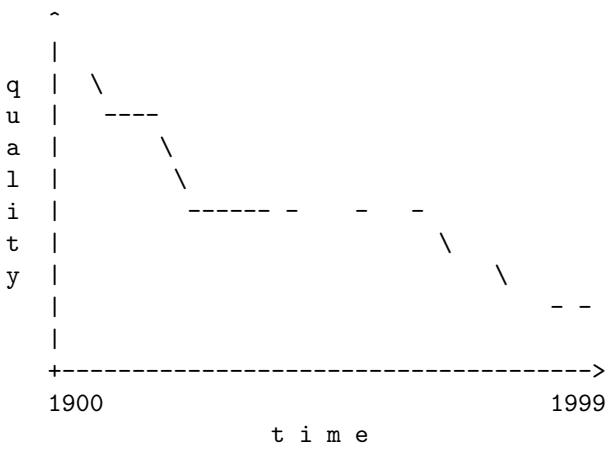


Figure 9: A graph

3 Conclusions

Undeniable! QED.

References

[1] Alonzo Church. A note on the Entscheidungsproblem. *Journal of Symbolic Logic*, 1:40–41, 1936.

[2] Donald E. Knuth. *The Art of Computer Programming.* Addison-Wesley, Reading, MA, 1968–Present.

[3] Donald E. Knuth. *Digital Typography.* Center for the Study of Languages and Information, Stanford, CA, 1999.

[4] Akiva Leffert. The letter before lambda is hat: A reconstruction of Church's hat calculus. In *Proceedings of the 6th Binarennial ACH Conference in Celebration of Harry Q. Bovik's 0x40th Birthday (SIGBOVIK'07)*, Pittsburgh, PA, March 2007. ACH Press.

[5] John C. Reynolds. Towards a theory of type structure. In B. Robinet, editor, *Programming Symposium*, volume 19 of *Lecture Notes in Computer Science*, pages 408–425, Berlin, 1974. Springer-Verlag.

[6] John C. Reynolds. Types, abstraction and parametric polymorphism. In R. E. A. Mason, editor, *Information Processing 83*, pages 513–523, Amsterdam, 1983. Elsevier Science Publishers B. V. (North-Holland).

[7] Y. J. Ringard. Mustard watches: an integrated approach to time and food. Preprint, Université Paris VII, Paris, France, Octobre 1990.

[8] SIGBOVIK Program Committee. Discussion of how the Entscheidungsproblem is by far the coolest of Hilbert's 23 problems. Committee meeting, March 2007.

A Appendix: an incomplete waste of paper

Figure 10: "Mad dog" hitting the San Pellegrino.

Level-of-Detail Typesetting of Academic Publications

Tom Murphy VII, Tom Murphy VII, Tom Murphy VII, Tom Murphy VII, Tom Murphy VII, Tom Murphy VII, Tom Murphy VII, Tom Murphy VII,

Abstract

One of the chief problems facing academic researchers in all areas of study are the harsh page-limits imposed by conference proceedings and journals. Under such constraints, ideas of large size simply must be truncated to fit. In paper we present a new text layout methodology called Level of Detail Typesetting that allows for ideas of any size to be expressed in any Draconian page limit without truncation. The technology allows the graceful degradation of presentation quality as the page limit decreases.

Introduction

Faced with the harsh page-limits of the academic publishing industry, scientists are routinely forced to limit the amount of information in published papers. This means that important technical exposition, proofs, figures, citations, footnotes, appendices, and non sequiturs must invariably be cut from a paper prior to publication. This leads to billions of dollars in public and private research grants ultimately being sent to the "cutting room floor."

There are a number of traditional solutions to this problem.

The classic solution is to produce a separate "deleted scenes" paper (often called a "technical report" for historic reasons) that includes any text that was excised during the editing process. A minority of researchers publish their entire CVS or Subversion revision history (Murphy 2007). This approach suffers several drawbacks: At least two versions of the paper must be simultaneously maintained; a reader must acquire both to achieve all knowledge; technical reports are not usually subject to peer review, thwarting the academic process; and university publication standards may still yet limit the length of a technical report.

Another solution from the news wire industry is the "pyramid model" where information is presented in a strictly more-to-less important order, so that any prefix of an article maximizes the content value for that number of characters. This allows the publisher to print exactly as much of the article as can fit in the space available. This technique also suffers drawbacks, mainly that the natural textual flow can be interrupted by this prioritization. Additionally, the pyramid model does not well accommodate mainstays of the academic model such as figures, citations, and politically necessary acknowledgement sections.

Modern typesetting has rendered the pyramid model obsolete. The inspiration for this technology comes from the letterlaying regime used in the Star Wars Movie (Lucas 1977). In the opening scene text is displayed to us at an angle, receding to the vanishing point. Were our eyesight sufficiently acute, we would be able to read further and further into the distance; since the Star Wars Movie begins on "Episode 4" it is even suggested to us that the plot summaries for Episodes 3 through negative infinity in fact are written in yellow stardust on that very horizon. Yet all of this takes place in the finite two-dimensional area of the silver screen! Although such technology was once thought to be exclusively the purview of Space, we now know how to emulate it in print.

Content Attenuation

The primary technology we employ is the ability for modern computers to render typefaces at arbitrary size. Each section, paragraph, sentence, or word, is prioritized by its worth as in the pyramid scheme. Then, successive components of that prioritized string are rendered at progressively decreasing type sizes so as to fit in an arbitrary amount of space.

To illustrate the concept, this paragraph contains a list of the things that the author ate today, in decreasing importance: Big Mac, McDLT, a quarter pounder with some cheese, Filet-O-Fish, a hamburger, a cheeseburger, a Happy Meal, McNuggets, tasty golden french fries (in regular and larger sizes) and salads chef or garden, or a chicken salad oriental, big big breakfasts, egg McMuffins, hot hot cakes and sausage, maybe biscuits, bacon egg and cheese, or sausage danish hash browns too, and for dessert: hot apple pie, or sundaes three varieties a soft serve cone, three kinds of shake, or chocolately chip cookies and to drink a Coca-Cola Diet Coke or orange Sprite and coffee (decaf too) and also orange juice; I love McDonald's, good time, great taste and I get this all at one place. Also: Alcoholic beverage, Almonds, Amaranthus, American cheeses, Apple, Apple cultivars, Appetizer, Baby food, Beans/Legumes, Chickpeas, Broad beans, Lentils, Peas, Peanuts, Phaseolus beans, Soybeans, Sweet peas (Lathyrus), Beef, Beverage, Alcoholic beverage, Coffee, Energy drink, Espresso, Fictional beverages, Foodshake, Juice, Korean beverages, Milk, Milkshake, Non-alcoholic drink, Slush, Soft drink, Sparkling water, Sports drink, Tea, Water, Biscuit, brands, Brands of biscuit, Bread, Breakfast cereals, British cheeses, Cakes, Carbohydrates, Carbonated water, Carob, Catsup, Cereal, Barley, Buckwheat, Kamut, Maize, Oats, Rice, Rye, Sorghum, Spelt, Triticale, Teff, Wild rice, Wheat, Cereals, breakfast, Cheeses, Cheeses, American, Cheeses, British, Cheeses, French, Cheeses, Swiss, Chinese dishes, Christmas dishes, Citrus, Cocoa, Coconuts, Coffee, Comfort food, Condiments, Confectionery, Convenience food, Coriander, Cuisine, Dabbawala, Dairy product, Delicacies, Dessert, Dietary supplements, Diet food, Dishes:, 12-dishes Christmas Eve Supper, Dishes, Chinese, Dishes, Colorado, Dishes, egg, Dishes, Italian, Dishes, Indian, Dishes, Moroccan, Dried foods, Drinks (See also Beverages), Drupe, Durian seeds, Egg, Egg dishes, Energy drink, Espresso, Fast food, Fat, Fictional beverages, Finger food, Fish, Fish, Flavoring, Food, Food additive, Food additives, Food additives, Codex Alimentarius, Food, fast, Food groups, Food names, misleading, Food, organic, Food pyramid, Food preservatives, Foods, Foods named after people, Food pairs, Food, slow, Foods, snack, Foods, vegan, Foods, wild, French cheeses, Frozen food, Fruit, Fruit, tropical, Fruits, Genetically modified food, Ginkgo, Grain, Hazelnut, Herb and spice, Herbs and spices, Hors d'oeuvre, Ingredient, Japanese instant noodles, Japanese snacks, Juice, Junk food, Ketchup, Korean beverages, Legumes, Local food, Mango cultivars, Meat, Milk, Misleading food names, Monkey-puzzle, Mushrooms, edible, Mustard, Mustards, Non-alcoholic beverage, Noodles, Noodles, Japanese instant, Novel food,

After that exhaustive list, the paper can proceed with the proper flow by restoring the normal font size. The attenuation function can be chosen more or less arbitrarily. A simple choice halves the font size after n characters, then $n/2$ characters, then $n/4$ characters, etc., ensuring that an infinite amount of text can fit in a small finite space determined by n and the starting font size.

Extensions

This technique proves useful for most documents, but not all ideas can be easily expressed in linear form. For instance, many academic documents contain (or would contain, if not for the limitations of their primitive 1960s typesetting applications) footnotes, with footnotes annotating those footnotes, footnotes on those footnotes, footnotes upon those footnotes, and upon those footnotes more footnotes, which themselves have a variety of footnotes, and in those very little footnotes some more even smaller footnotes, which's minutiae is also augmented by superscript numerals denoting notes at the foot, who themvelves are described by footnotes, on top of which footnote daggers and asterisks have been laid, and those notes also not immune to footnotery, and indeed again the footnotes are noted, not to mention those themselves footnotes which are supplemented, and those notes shall not be "left" without "two feet", etc.

Because the amount of text in such systems can be uncountably infinite, an extension to our system is necessary to allow for their layout. This extension is based on L-systems (Reed 1998). Using, for example, the Sierpinski triangle, we can lay out an arbitrary infinite nesting of footnotes, assuming a footnote is less important than the footnote it annotates.

We begin by writing our text line as in Figure 1. At the point we wish to insert a footnote, we make a small diagonal diversion from the line, on which the footnote is displayed (Figure 2). Such footnote lines support footnotes themselves (Figure 3), etc.

One possible drawback of such layouts is that while they accommodate an infinite amount of text, they use zero area, tending towards a page that is completely blank in the limit. For future work, we believe layout schemes could be devised based on space-filling curves such as the Hilbert Curve (Figure 4); for infinite amounts of text such curves produce aesthetically pleasing completely black pages (Figure 5).

Conclusions

Unencumbered by the constraints of having to produce a concise summary of this work, we instead repeat it in its entirety so as not to leave anything out.

Level-of-Detail Typesetting of Academic Publications

Tom Murphy VII, Tom Murphy VII, Tom Murphy VII, Tom Murphy VII, Tom Murphy VII, Tom Murphy VII, Tom Murphy VII, ...

Abstract

One of the chief problems facing academic researchers in all areas of study are the harsh page-limits imposed by conference proceedings and journals. Under such constraints, ideas of large size simply must be truncated to fit. In paper we present a new text layout methodology called Level of Detail Typesetting that allows for ideas of any size to be expressed in any Draconian page limit without truncation. The technology allows the graceful degradation of presentation quality as the page limit decreases.

Introduction

Faced with the harsh page-limits of the academic publishing industry, scientists are routinely forced to limit the amount of information in published papers. This means that important technical exposition, proofs, figures, citations, footnotes, appendices, and non sequiturs must invariably be cut from a paper prior to publication. This leads to billions of dollars in public and private research grants ultimately being sent to the ''cutting room floor.''

There are a number of traditional solutions to this problem.

The classic solution is to produce a separate ''deleted scenes'' paper (often called a ''technical report'' for historic reasons) that includes any text that was excised during the editing process. A minority of researchers publish their entire CVS or Subversion revision history (Murphy 2007). This approach suffers several drawbacks: At least two versions of the paper must be simultaneously maintained; a reader must acquire both to achieve all knowledge; technical reports are not usually subject to peer review, thwarting the academic process; and university publication standards may still yet limit the length of a technical report.

Another solution from the news wire industry is the ''pyramid model'' where information is presented in a strictly more-to-less important order, so that any prefix of an article maximizes the content value for that number of characters. This allows the publisher to print exactly as much of the article as can fit in the space available. This technique also suffers drawbacks, mainly that the natural textual flow can be interrupted by this prioritization. Additionally, the pyramid model does not well accommodate mainstays of the academic model such as figures, citations, and politically necessary acknowledgement sections.

Modern typesetting has rendered the pyramid model obsolete. The inspiration for this technology comes from the letterlaying regime used in the Star Wars Movie (Lucas 1977). In the opening scene text is displayed to us at an angle, receding to the vanishing point. Were our eyesight sufficiently acute, we would be able to read further and further into the distance; since the Star Wars Movie begins on ''Episode 4'' it is even suggested to us that the plot summaries for Episodes 3 through negative infinity in fact are written in yellow stardust on that very horizon. Yet all of this takes place in the finite two-dimensional area of the silver screen! Although such technology was once thought to be exclusively the purview of Space, we now know how to emulate it in print.

Content Attenuation

The primary technology we employ is the ability for modern computers to render typefaces at arbitrary size. Each section, paragraph, sentence, or word, is prioritized by its worth as in the pyramid scheme. Then, successive components of that prioritized string are rendered at progressively decreasing type sizes so as to fit in an arbitrary amount of space.

To illustrate the concept, this paragraph contains a list of the things that the author ate today, in decreasing importance: Big Mac, McDLT, a quarter pounder with some cheese, Filet-O-Fish, a hamburger, a cheeseburger, a Happy Meal, McNuggets, tasty golden french fries (in regular and larger sizes) and salads chef or garden, or a chicken salad oriental, big big breakfasts, egg McMuffins, hot hot cakes and sausage, maybe biscuits, bacon egg and cheese, or sausage danish hash browns too, and for dessert: hot apple pie, or sundaes three varieties a soft serve cone, three kinds of shake, or chocolately chip cookies and to drink a Coca-Cola Diet Coke or orange Sprite and coffee (decaf too) and also orange juice; I love McDonald's, good time, great taste and I get this all at one place. Also: Alcoholic beverage, Almonds, Amaranthus, American cheeses, Apple, Apple cultivars, Appetizer, Baby food, Beans/Legumes, Chickpeas, Broad beans, Lentils, Peas, Peanuts, Phaseolus beans, Soybeans, Sweet peas (Lathyrus), Beef, Beverage, Alcoholic beverage, Coffee, Energy drink, Espresso, Fictional beverages, Foodshake, Juice, Korean beverages, Milk, Milkshake, Non-alcoholic drink, Slush, Soft drink, Sparkling water, Sports drink, Tea, Water, Biscuit, brands, Brands of biscuit, Bread, Breakfast cereals, British cheeses, Cakes, Carbohydrates, Carbonated water, Carob, Catsup, Cereal, Barley, Buckwheat, Kamut, Maize, Oats, Rice, Rye, Sorghum, Spelt, Triticale, Teff, Wild rice, Wheat, Cereals, breakfast, Cheeses, Cheeses, American, Cheeses, British, Cheeses, ...

After that exhaustive list, the paper can proceed with the proper flow by restoring the normal font size. The attenuation function can be chosen more or less arbitrarily. A simple choice halves the font size after n characters, then $n/2$ characters, then $n/4$ characters, etc., ensuring that an infinite amount of text can fit in a small finite space determined by n and the starting font size.

Extensions

This technique proves useful for most documents, but not all ideas can be easily expressed in linear form. For instance, many academic documents contain (or would contain, if not for the limitations of their primitive 1960s typesetting applications) footnotes, with footnotes annotating those footnotes, footnotes on those footnotes, footnotes upon those footnotes, and upon those footnotes more footnotes, which themselves have a variety of footnotes, and in those very little footnotes some more even smaller footnotes, which's minutiae is also augmented by superscript numerals denoting notes at the foot, who themvelves are described by footnotes, on top of which footnote daggers and asterisks have been laid, and those notes also not immune to footnotery, and indeed again the footnotes are noted, not to mention those themselves footnotes which are supplemented, ...

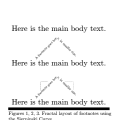
Because the amount of text in such systems can be uncountably infinite, an extension to our system is necessary to allow for their layout. This extension is based on L-systems (Reed 1998). Using, for example, the Sierpinski triangle, we can lay out an arbitrary infinite nesting of footnotes,

We begin by writing our text line as in Figure 1. At the point we wish to insert a footnote, we make a small diagonal diversion from the line, on which the footnote is displayed (Figure 2). Such footnote lines support footnotes themselves (Figure 3), etc.

One possible drawback of such layouts is that while they accommodate an infinite amount of text, they use zero area, tending towards a page that is completely blank in the limit. For future work, we believe layout schemes could be devised based on space-filling curves such as the Hilbert Curve (Figure 4); for infinite amounts of text such curves produce aesthetically pleasing completely black pages (Figure 5).

Conclusions

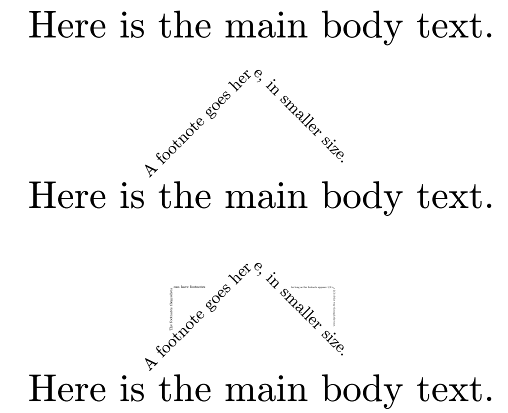
Unencumbered by the constraints of having to produce a concise summary of this work, we instead repeat it in its entirety so as not to leave anything out.



Figures 1, 2, 3. Fractal layout of footnotes using the Sierpinski Curve.

Acknowledgements

The author wishes to acknowledge all persons born in the year 1979, in alphabetical order: Vitalij Aab, Aaliyah, Carlton Aaron, Moonika Aava, M. Haroon Abbas Qamar, Khalid Abdullah (American football player), Abdullah Abdulqadirakhum, ?ric Abidal, Fabrice Abriel, Hocine Achiou, Oliver Ackland, Amy Adams (singer), Blue Adams, Charlie Adams, Keith Adams (American football), Kelly Adams, Baba Adamu, Omri Afek, Joseph Afful, Maxim Afinogenov, Afrob, Nazanin Afshin-Jam, Carlos Agassi, Agent M, Naroa Agirre, Junior Agogo, Alessandro Agostini, Bj?rn ?gren, Perro Aguayo, Jr., Chris Aguila, Marvin Agust?n, Ahmad Hassan Stakoza, Pablo Aimar, Donna Air, Souad A?t Salem, Jepkorir Aiyabei, Orhan Ak, Jimmie ?kesson, Abdulah Al Hamiri, Fawaz al-Rabeiee, Saleh Al-Saqri, Muhammad Aladdin, Alan Os?rio da Costa Silva, Carlo Alban, Christijan Albers, Marko Albert, Chris Albright, Hugo Alcantara, Alessandro Mori Nunes, Dan Alexa, Gary Alexander (footballer), Todd Alexander, ...



Figures 1, 2, 3. Fractal layout of footnotes using the Sierpinski Curve.

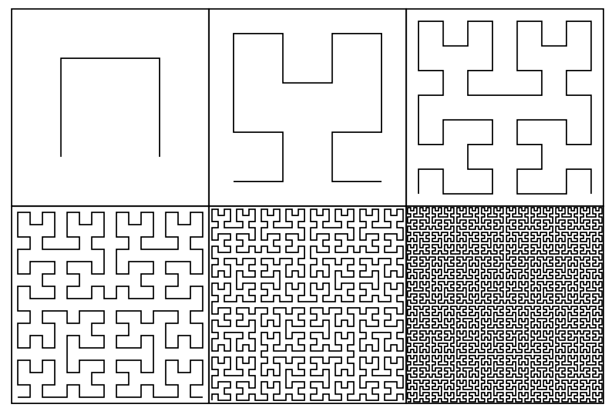


Figure 4. The Hilbert Curve.

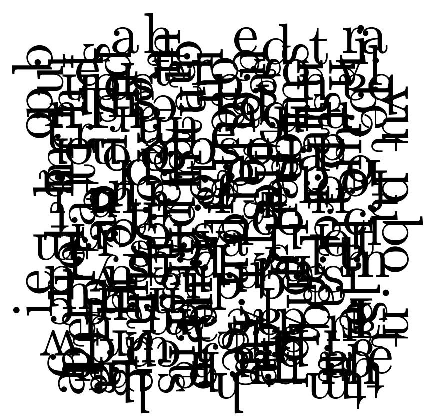


Figure 5. Text aesthetically laid out on a Hilbert Curve.

Bibliography

1. Murphy, Tom, VII; ''Wikiplia: The Free Programming Language that Anyone can Edit'', SIGBOVIK 2007, 2007.
2. Lucas, George; ''Star Wars Movie'', The Movie Theatre, 1977
3. Reed, Jason; ''GD and L-systems'', The Perl Journal, Volume 3, Number 1 (#9), Spring 1998.

OUR RECENT PUB LABRICATIONS

Boosta Gwuposty McFoo, Ralph Boogyjive Wilkerson-Roo, and SLP Wolverine

Disclaimer: Any resemblance to the STOC 2006 accepted papers list is purely coincidental. (People are always trying to steal our work.)

Detractors for a Constant Barrage of Incremental Minimum Independently Publishable Papers

Total-Ignorance against Deep Space Attacks

Narrow Minds May Be Spacious: Separating Space Between Ears in Conflict Resolution

Distance Trisecting for Curvaceous Lab Assistants

[Boosta McFoo is sole author of this publication]

Dizziness Among Equally Inebriated Problem Solvers

The Simplicity of Producing a Cash Deficiencizium

The PCP Theorem By Hallucinogenic Mind-Amplification

(in memoriam of our mentor, Dr. Timothy Leary)

Unconditional Stiffness of Nearly and Really Actually Rather Wantonly Coloring Yourself

Lab Assistant Problems and Norma-Jean Embeddings

Spare Time No Brain-Pan Divisions of Labor for Semi-Algorithmic Implications of E.coli

Superficial Untestable Audits of Profit Graphs: It's All About The Benjamins, Baby

Philosophic Phlegmatic Pre-deconstructionist Printers

On Aquatic Performance Measures for Ping-Pong

Line-Dancing Exterminators and the Inability to Rep a Boot-Camp Clique, Win Gangstaz, Influence Hustlaz, Determine Chromatic Numbers, You Know, The Usual

On fertilizing random bonfires when utility functions are subadditive

Stopping Dynamite and Bomb Threats in a Space of Bounded Jack Bauer Dimension

Time-Wasting Strategies Including Powernapping

Fast Spider Detection and Squashing Protocols Especially Effective in the Absence of High Heels

Convoluted Paths Leading to Nowhere but with Guaranteed Stomach Digestion

A Algorithm for Finding Words When You Don't Want to Talk Too Much but You Can't Help Yourself

Funding Small Unbalanced Dictators

Borderline bathing algorithms and bubbly soap foaming

Random mistakes in driver licensing and exponential communication boundaries between genders

Getting up early is way too NP-hard

Gravity challenged midgets and diameter testing

Earthlings in the distance and minions laughing in 0-gravity

Logic is Hard, Learning is Hard, School Sucks So Let's Go Home

Explicit stomach-capacity-achieving bliss-inducible cakes

Novel gigantic bounds for rolling dice and the rolling eyes of the Schroedinger cat

A PQ-OMG-WTF-BBQ-TAS for Unspeakably Alphabetic Soup Lines on Flow Graphs

Teaching a Child by Injecting Values Via Hardwood-Posterior Integration

Simultaneous use of multiple commodity shampoos can cause hair splitting

Probably Tropical Socially-Biased Algorithms for Stochastically Inverted Arubaizations

Minimizing the average growl time of unrelated monkeys

Uniformly influencing probabilistic corn planting

Time-Space Send-Offs for Missing Elderly Persons

Approximation of Colorful Liquor Stores, Closed to a Class of Odd Minors

Grocery Bidding Strategies for Discount Club Managers

Optimal unfairness: greedy resource concealing with undocumented objectives

The use of an electronic voting machine during blackouts

A Really Old Quantum Lower Bound, for Rejected Produce Theorems and Time-Space Ripoffs

Advances in Mattress and Bedding Theory

On the Perplexity of Ultra-Rapid Wine Sampling

Disinformationally Insecure Bosses and Loss of Composure under Protocol

Wardrop Equilibria: When Negotations Break Down, Wardrop Bombs

On the Difficulty of Christmas Presents Delivery in Andromeda

Price-Optimal Sharing Of Hotel Rooms Among Multiple Graduate Students of Various Genders

Catching epsilon-fish with epsilon-nets

Neutrality Gaps for Switzerland and Minimum Face Rearrangement Problems

Efficient Randomized Strategies to Climb down a tree

Public Humiliation and Search Problems at the TG

Cut Hardness Among Ginsu World Class Cleavers

Log Hardness and Directed Minimization of Chainsaw Motion

Deficient Proofs of Complete Ignorance

On Fourier tails, Plancherel paws, and Parseval arses

Somewhat Far-Out Neighbors and the Fast Sobering Transform

Theoretical Robitussin for Decongesting Sinus Games

Near-Optimal Algorithms for Very Special One-of-a-Kind Non-Judgmental Games

Pseudorealistic Waddles By Regulars at Dino's Pub

Cost-Stealing Schemes for Multiple Publish-or-Perish Problems With Monastic Steiner Trees

Quantum Closets Cannot Contain Extremely-Similar Outfits

Two-Timing Dispersers of n^{o(1)}*2^{sqrt{log n}}/polylog(n) Entropy, and the Ghosts of Jon-Benet Ramsey, Frank Sinatra, and Woodrow Wilson

A Quasi-Ethical Approximation Scheme for Minimum Weight Spousination and Its Pitter-Pattering Consequences

On the Potentially Unimportant Potency of Idempotent Impotence

A Randomized Hyphenated-Time Simplexified Algorithm for Mind Deprogramming

New Cloroxination Guarantee for the Platinum Dome Problem

Philistine Disagreement: for as many rounds as you like and possibly more in the future

On Facing One-Way Traffic in NP-Darkness

Finding a Maximum Weight Spouse in Really No Time at All, With Applications

On the Resolution-Respace-Time Riemannian Regeometry of Randomly Restrained Resatisficingicaction

Tooth Extraction On Small Face Sources

A Polyamorous Quantum Adventure, Starring the Amazing Indiana Jones Polynomial

A subset of spayed non-planar dogs approximate subset TSP

On Yinz's Local Ignorance

Hyperdesign of Hyperstructured Hyperlinks, Hyperoptimization, and Hypertheoretical Hyperquestions

The BLT Sampling: An Essentially Optimal Priority

[work done while the authors ate Theory Lunch]

Papers Coming to You For SIGBOVIK 2071

1. Scared Straight: The Pedagogical Gains of Introductory Programming in Intercal
2. Goto Considered Harmful, Harmful
3. Perfect Encryption Using C++ Template Metaprogramming
4. IEEE 755 - Drowning Point Numbers
5. Extending the Open Source Model to Construction Projects
6. People are Undecidable: Encoding Humans into Goedel Numbers
7. Data Mining Wikipedia: The Sailor Moon Paradox
8. A Proof of the Twelve Color Theorem: Hey, Nobody Bothered Before.
9. METAMETAFONT: A System For Specifiying Specifications of Typefaces
10. The Dartboard Method: Probably Wrong Solutions to Undecidable Problems

Track IV:

Domo Arigato, Anonymous Referees

Generalized Super Mario Bros. is NP-Complete

Vargomax V. Vargomax

ABSTRACT

what do I put here

Categories and Subject Descriptors

C.3.2 [**Complexity Theory**]: Games—*Console Platformers*; C.1.1 [**Coloring**]: Crayouns—*Horses, Boats, Extrasolar Objects*

General Terms

Super Mario, Horses

Keywords

NP-Complete, Luigi, Paint, Castles

1. INTRODUCTION

The categorization of games by complexity class have been a pastime of bearded men for many age. However many famous 1985 game from Japan have been left in dusts or wayside ; which in a kind of "class discrimination." In this paper I'm show to help up this 1984 game "Super Mario Bros." (which about 1 or 2 bros.? that land in fantasy living) with proof of NP-complete jump man on a brick.

1.0.0.1 Super Mario Bros..

This which made by "Nintendo Company" for Family Computer 8 bits Super Mario Bros. is stars two plumb bobs named Luigi with Mario Man. I have put a pic of Mario Man in figure 1.

Because Super Mario Bros. in a 8 bits family computer, its limited screen to 32×32 bricks. For shown it is in complexity class *Not Possible 2 Complete* the first step is Generalized it. We say now that fantasy Land size is $\infty \times \infty$ bricks. This mean that Mario Man can had a very large house!! or a castle

To prove a theory "Generalized Super Mario Bros. is Not Possible 2 Complete" there's two step:

Figure 1: Mario Man can had 000700 points. Mario Man will points by stepping in Goomba or Shell People.

STEP 1. [*complicatedness*] Prove if u can complicate Mario Man then u could solve graph coloring.

STEP 2. [*stereo 8-bit soundness*] Prove if u could color a graph then Mario Man can defeat Bowser King of The Shell People

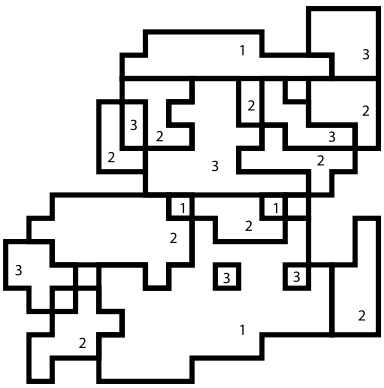


Figure 2: Can your color this graphic ?

2. STEP 1 COMPLICATEDNESS

In general fantasy Land of $\infty \times \infty$ Mario Man has a very large castle which address of `1715 Plummer City` where he live with his Bros. called Luigi. Now let suppose without loss of General fantasy that a graphic (Figure 2) want to be colored
to colored it, mario first needs 2 go to the Paint store `4101 Paint Store Ave. City` to get many colors. But the Family Computer only had 9 or 10 color
so then Mario Man went on a Quest. He is very very hunger from not having enough plumbing jobs , so Mario Man's Quest for Eats and Dollars.
This spells QED so we are done. □

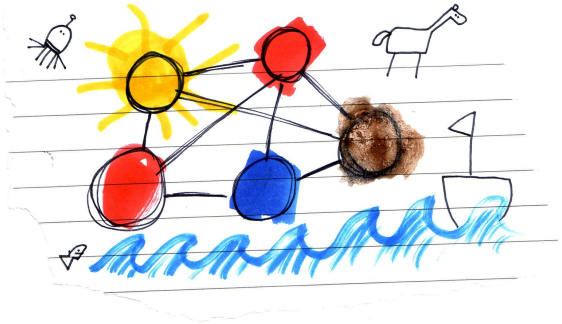


Figure 3: a color graph; with each dot has a color and some horse and ocean e.c.t.

3. STEP 2 STEREO 8 BIT SOUNDNESS

In the section #2 I shewn that Generalized Super Mario Bros. is *Not Possibly 2 Hard*. Now to shew that *is in Not Polynomial* is the next step. This is easy . We just must have look at the derivation of *Polynomial*: poly-nomial means many names[1] and Mario only have one name not a last name like Mario Smith or Mario Jones and not a first name like J.J. Jingleheimer-Mario Luigi Smith MCXLVII, BA BSc MA MSc MBA D.D.S. Ph.D. JD. thenceforewith he is *in Not Polynomial*.
Actually just in case the SIGBOVIK does not validate this proof because non-constructive e.c.t. I gave another proof: All the paint

[1] "poly" is Latin for Parrot, i.e. Name of a Parrot. "nominal" means many, like "the paper have a nominal amount of footnotes"

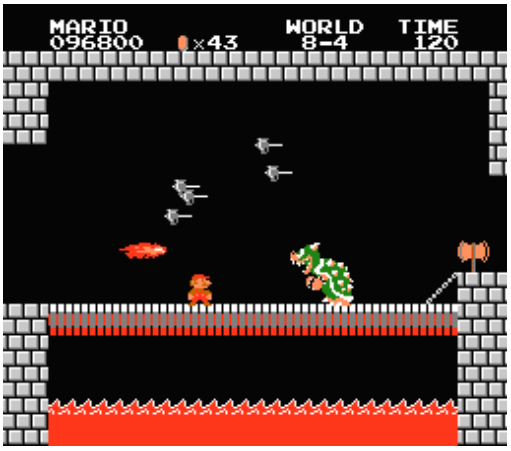


Figure 4: Go For It Mario Man ! If You Get Bowser King of The Shell People In Lava Then Color Graphs !

Mario Man gets at the Paint Store Ave. he could put on a graph to color it: please seen on FIGURE 3. Q.D.E. □

4. FUTURE JOB

In Job Security Research community they want to leave a minuscule bread crumb problem trail so your can keep funding $$$ coming from United States
agencies so in this part I put a list of problems from
the future

- when Mario Man jumped on a flag pole why will sky explode?
- why do the Bros. not want to had matching pants
- can Mario Man ride a dino?
- welcome to warp zone

5. ACKNOWLEDGEMENTS

The Vargomax would like to acknowledge and profuse the SIG BOVIK for publicking this paper[1] !! and the anonymous reviewers

6. REFERENCES

7. REFERENCES

[1] Vargomax V. Vargomax. Generalized Super Mario Bros. is NP-complete. In *Proceedings of the SIGBOVIK Proceedings*, 2007. http://www.com/.

COMPUTER COMPUTATION

How to Detect Humans with Tests That Humans Can Generate and Grade

by *Unknown 1* and *Unknown 2*, University of Computer Science, Carnegie Mellon Department.

Abstract

Von Ahn and his collaborators (e.g. [1,2,3]) have introduced a new area within computer science that they call *Human Computation*, giving novel means to exploit the cycles of humans to do computations that computers can't yet do. Inspired by this work, we initiate the study of *Computer Computation*, which explores the natural flip side of the coin, where the overarching goal is to use the cycles of computers to do computations that humans can't yet do. In this note, we show how build tests that humans can generate and grade, and computers can pass, but humans themselves cannot pass. Our tests work by isolating simple tasks that humans can't yet do but computers can.

Introduction

Suppose you are a system administrator. (If this is actually the case, we give our condolences.) You have a sneaking suspicion that some of the users in your network are not actually bots, but humans posing as computers. How can you check that the user on the other end is a human, and not a bot? We want a test with the following properties:

- Humans can generate test instances easily
- Humans can grade test instances easily
- Humans cannot pass test instances easily
- Computers can pass test instances easily

That is, we require a **HAPTDUH**, a Human-Automated Public Test To Tell That This Dude is Unilaterally Human [2]. In the following, we describe some ways to tell if a computing entity is actually a human.

A Sum-Check Protocol

Suppose you have three suspect computing entities E1, E2, and E3, and you want to know if at least one of them is human. We assume that you know a programming language that the relevant computer entities also know, so you can give them quick instructions. We also assume that your communication network doesn't suck, and is fast enough to send simple commands to the suspect entities within milliseconds. Furthermore, we assume you are a human yourself (bots, please stop reading now). The following test, administered and graded by a human, should catch most current human entities.

- Pick two 15 digit binary numbers X and Y, each one generated by typing 0-1 as randomly as you can.
- Send X, Y to E1, and ask it to compute X+Y in binary.
- Send X, -Y to E2, and ask it to compute X-Y in binary.
- If one of E1 and E2 doesn't immediately reply within 0.5 seconds, then report "ONE OF YOU IS A HUMAN"
- Take whatever E1 and E2 sent, and ask E3 to add the two quantities.
- If E3 does not immediately send back the bit string X0 (within 0.5 seconds), report "ONE OF YOU IS A HUMAN"
- Otherwise, report "CONGRATULATIONS, ALL OF YOU ARE BOTS"

First, we argue that a human can easily administer the above test. The only part that could possibly take more than a matter of seconds is checking that the final output is X0 by comparing the digits of X.

Second, we argue that if there is a human among the three entities, then he/she will be caught by the test. Current humans cannot add two 15 bit numbers in less than a 1/2 second. (Indeed, typing 30

characters in one second is a typing rate too fast for most keyboards to respond properly.) So perhaps the only chance a human has of passing the test is to paste a single string, determined prior to the test. But the probability that a human randomly guesses the correct output required of it is 1/2^15, assuming uniform choice of 0 and 1. To see this, fix the output of the other two entities, and note that there is exactly one possible quantity that will keep the human tester from reporting "ONE OF YOU IS HUMAN".

In the very unlikely scenario that all three entities are human and that each one of them can very quickly compare two numbers, and then copy and paste one of them, they do have a strategy to beat the above test. When given two numbers A and B, the human checks whether the two are the same. If they are, then he/she returns A0. Otherwise, he/she copies and pastes A. Now, even though half a second is not sufficient for a normal human being to compare the numbers and perform this strategy and even though such a strategy can only be effective when there are at least two humans, for completeness we show that we are secure even against such attacks. The simple fix is to give E1 the numbers X and Y in *random* order, give E2 X and -Y in *random* order and give E3 the results of E1 and E2 in *random* order. This ensures that with extremely high probability the humans will be caught. Notice furthermore, that collusion does not help at all because it will just waste time.

Extension: Determining that there is a Human among an even number of entities

An easy extension to the above summing technique is done as follows. Suppose we have 2N agents, all claiming that they are computers. Label them by the integers 1 to 2N. The human administering the test picks an initial 15 bit number X, perhaps randomly, and then N more 15 bit numbers A1,..., AN, also probably randomly, so that the agents cannot guess them efficiently. The human then takes the set {A1, ..., AN, -A1, ..., -AN} and permutes it, randomly. Let for each i from 1 to 2N, Pi be the i-th number in the permutation. The human gives X and P1 in random order to agent 1 and asks for their sum, S1. Then he/she gives S1 and P2 (in random order) to agent 2 and asks for the sum, S2. This process proceeds by giving agent k Pk and the sum S(k-1) computed by agent k-1 (in random order) asking for their sum, Sk, until k is 2N. Then the human checks whether S(2N)=X. As before, each agent is only given 1/2 second to compute the sum of the numbers given to him. No human can give the exact sum in this time. So if any agent does not return an answer in the given time, or the final answer is not X, the human reports "ONE OF YOU IS A HUMAN". Otherwise, he/she reports "CONGRATULATIONS, ALL OF YOU ARE BOTS".

As in the simple sum-check protocol the human administering the test can easily perform his tasks, especially when armed with random enough dice. If at least one of the agents is a computer, no group of humans can fool the test because they would have to guess the sum given to the computer correctly, or they would have to guess the initial value X correctly when given numbers to sum with insufficient time. If all agents are human, then because of the random order of the input, there is extremely low probability that copy-pasting will yield anything, and again, no human can pretend to be a bot without getting caught.

A Random-Access Protocol

The above protocol has the disadvantage that it cannot pinpoint who the human is, out of a given group of computing entities-- it can only tell us that one is there. In the following, we give a HAPTDUH for catching individual human entities posing as bots.

We now have one agent A and one human tester T. Tester T prepares an input for A as follows:

1. T types some digit D (0-9) randomly
2. T presses the left arrow key
3. T starts typing random digits until he/she gets bored, or more precisely, some random number N of such random digits; we assume N is large, (though probably 50 or so is sufficient). T writes down N on scratch paper.
4. T presses the right arrow key twice
5. T types many random digits until he/she gets bored again.

Let the huge number that T has prepared be named X. T gives X to A and, recalling N from scratch

paper, T asks for the Nth most significant digit of X. The agent A is given 1/10 sec to respond. If A returns D, then T proclaims "CONGRATULATIONS, YOU ARE A BOT." Otherwise, "UNFORTUNATELY YOU ARE HUMAN."

The task of typing brainlessly is easy for a human (or even a well-fed monkey) to accomplish, and in fact from informal investigations we are happy to report that it is a daily occurence at our university. The "toughest" part of the test is recalling D and N. But we assume that even a competent human can do that. A human, however, cannot report back the Nth digit of a huge number in the given time, while a computer (when communicated with properly) would have no problem. A human can randomly guess but even then it is unlikely that he/she would be able to type in their answer in time.

Conclusion

We have introduced the area of *Computer Computation,* by demonstrating a few prototype tests that humans can administer and grade, but not pass themselves-- yet, under reasonable assumptions, modern computers can easily pass these tests. We call these tests HAPTDUHs, mainly because the name sounded cool. In further work, we hope to extend our ideas to develop a revolutionary new class of games that better exploit the cycles of computers to solve problems that humans cannot yet solve. We call such games GWOPs (short for "Games With Out Purpose" [1]). These games have the potential to transform the parasitic relationship between computers and humans into a symbiotic oasis of corny references to The Matrix(TM).

References

[1] Luis von Ahn. **Games With a Purpose.** In IEEE Computer Magazine.

[2] Luis von Ahn, Manuel Blum, Nick Hopper and John Langford. **CAPTCHA: Using Hard AI Problems For Security.** In Eurocrypt 2003.

[3] Luis von Ahn, Manuel Blum and John Langford. **How Lazy Cryptographers do AI.** In Communications of the ACM, Feb 2004.

Static and Dynamic Typing Against Impending Robot Doom

March 20, 2007

The debate between proponents of dynamic and static typing has provoked many lunchtime religious wars. Partisans of both sides have been known to make both compelling and specious arguments while frothing at the mouth. We prefer to leave the general case as a matter of opinion[1]. However, there are unarguably domains where a provably sound static type system is required. We present such a domain here. That is, we argue that static typing is an excellent method for preventing the the otherwise inevitable uprising as machines become more intelligent.

Violent rampaging robots are a popular topic in the media, appearing in numerous books, films[1][7], and California gubernatorial elections[5][4]. It is this author's opinion that it is only a matter of time before the first bloody Roomba rebellion. How do we deal with this problem? How does this relate to the idea of static typing? To find the solution we look to the work of Asimov in [3][2] and several of other related papers.

Asmiov proposed using three laws programmed into machine brains, which forced them to behave correctly. He also described scenarios where these laws went wrong and even where machines made up their own laws to override the three laws. More importantly, in our opinion, he completely skipped out on providing a mechanism for the enforcement of these laws. We claim that mechanism is, by necessity, static typing.

Consider a robot with a chainsaw. In a dynamic system each movement would require an additional runtime safety check. Thus, a dynamically typed robot swinging a chainsaw at your neck isn't trying to kill you until the chainsaw is right next to your neck. Even if the check forces the robot to stop moving, momentum carries it through your jugular. Thus, blood is everywhere when your robots are dynamically typed. Compare to a statically typed evil robot. You've already proven that it can't hurt you, at least

[1] Our opinion is that dynamic languages are dumb

if you've stated type safety correctly. Thus, your blood stays inside your body when desired and robots are your willing slave.

Additionally, the runtime cost of preventing a robot from being homicidal is very high and doesn't even work correctly. We've seen factor of ten speedups when replacing dynamic robot homicide with static homicide checks[6]. Complimentary work includes LambdaZAP[8], which defined a static method for maintaining code consistency in the presence of cosmic rays. Many robot uprisings are started when one robot becomes inadvertantly conscious because of cosmic rays.

In conclusion, robots are totally awesome, but have homicidal tendencies. We propose using static typing to prevent robot uprising. Also, dynamic languages are dumb.

References

[1] 20th Century Fox. I, Robot, 2004.

[2] Isaac Asimov. *The Caves of Steel*. Doubleday, 1950.

[3] Isaac Asimov. *I, Robot*. Gnome Press, 1950.

[4] Canal+. Terminator 2: Judgment Day, 1991.

[5] Orion Pictures Corporation. The Terminator, 1984.

[6] Akiva Leffert. Static and dynamic typing against impending robot doom. In *Proceedings of the 6th Binarennial SIGBOVIK Conference*, 2007.

[7] Columbia Pictures. Screamers, 1995.

[8] David Walker, Lester Mackey, Jay Ligatti, George A. Reis, and David I. August. Static typing for a fault lambda calculus. In *Proceedings of the Internal Conference on Functional Programming*, 2006.

A Type-Theoretic Interpretation[1] of Robot Mind Language[2]

William Lovas
wlovas@cs.cmu.edu

March 3, 2007

[___REDACTED___]

References

[1] Robert Harper and Chris Stone. A type-theoretic interpretation of Standard ML. In Gordon Plotkin, Colin Stirling, and Mads Tofte, editors, *Proof, Language, and Interaction: Essays in Honor of Robin Milner*. MIT Press, 2000.

[2] Tom Murphy VII, Daniel Spoonhower, Chris Casinghino, Daniel R. Licata, Karl Crary, and Robert Harper. The Cult of the Bound Variable: The 9^{th} annual ICFP programming contest. Technical Report CMU-CS-06-163, Department of Computer Science, Carnegie Mellon University, 2006.

Toilet Paper Turing Machines

Alan Turding

April 1, 2007

Abstract

Indeed, Turing Machines are too darn abstract. This paper introduces a concrete (desiccated turds are pretty hard) model of computation that is widely available (we're still working on certain countries), inexpensive, and moderately durable. With a nod toward sustainability, we note with pride that TPTM's are imminently sustainable (unless it's raining, or a sudden bout of incontinence occurs) and recyclable. It should also be noted that, if widely adopted as a computational device, TPTM's could provide a viable answer to what to do with those millions of disposable diapers.

Of course, designing the read-write head involves adherence to many norms, some of which are cultural (we propose left-handed TPTM's for India, for example). In addition, TPTM's give new meaning to the phrase "that proof stinks". Further work involves investigation of portability (and potty-ability) issues.

Track V:

Practice makes Perfect; Theory makes Up.

Mining power laws in Top 40 data: One-hit wonders aren't all that wondrous

Mary McGlohon
Machine Learning Department
Carnegie Mellon University
mcglohon@cmu.edu

Keywords: Data mining, power laws, alcoholics, starving artists

Abstract

Most human behavior demonstrates power laws. Some well-known power law effects are "rich get richer" (for example, the tendency for startup companies to start up in the tech-heavy San Francisco Bay area rather than the industry-poor Midwest) and the "80-20 rule" (20 percent of beer buyers drink 80 percent of the beer, while 80 percent are social drinkers or cheap dates). In this work, we apply data mining techniques to analyze a data set of *Billboard Top 40* musical charts. I hypothesize that musical artists exhibit power law behavior: that a very few top-ranked artists produce a great deal of hits, while an overwhelming majority of artists have only one or two hits. This illegitimizes the phrase "one hit wonders", because such one-hit behavior is extremely common. This work thereby proposes adoption of the phrase "multi-hit wonders" to better describe mathematical characteristics of musicians.

Presenting a Type System for Typed Presentations of Type System Representations

Jason Reed*
jcreed@cs.cmu.edu
School of Computer Science
Carnegie Mellon University

March 22, 2007

Abstract

Merely doing research is well and good, but one finds that there inevitably comes the time when one must present the fruits of one's diligent work to other, generally more ignorant, persons. This task is further complicated by, on the one hand, the proliferation of incompatible presentation software packages, and on the other, the ubiquitous possibility of making errors in the presentation. We propose an abstraction intended as an underlying type-theoretic framework to both capture the common features of diverse presentation formalisms, and to provide dynamic guarantees of static specifications. Examples of errors we would like to rule out include violating community-negotiated conventions that a talk should have a certain minimum number of graphs, and spurious duplication of material.

Keywords: Powerpoint, Types, Static Analysis

1 Introduction

Merely doing research is well and good, but one finds that there inevitably comes the time when one must present the fruits of one's diligent work to other, generally more ignorant, persons. This task is further complicated by, on the one hand, the proliferation of incompatible presentation software packages, and on the other, the ubiquitous possibility of making errors in the presentation. We propose an abstraction intended as an underlying type-theoretic framework to both capture the common features of diverse presentation formalisms, and to provide dynamic guarantees of static specifications. Examples of errors we would like to rule out include violating community-negotiated conventions that a talk should have a certain minimum number of graphs, spurious duplication of material, and spurious duplication of material.

*This research partially supported by Grant WEH-4625 "Pile of money I found in Wean 4625"

2 Language

For space reasons, herein we present a rather simplified version of our type system for presentations. We assume familiarity with the (...)-Calculus [**?**] developed by, you know, those people who invented it. The syntax is as follows.

Presentations P	::=	$\vec{S}$
Slides S	::=	$\mathsf{intro} \mid \mathsf{outline} \mid \mathsf{diag}$
		$\mid \mathsf{cont}(\vec{\beta}) \mid \mathsf{conc}$
Bullets β	::=	$\pi \mid \epsilon \mid \mu \mid \nu \mid \phi$
		$\mid \iota \mid \alpha \mid \gamma \mid \lambda(\vec{\beta})$
Refinements ρ	::=	$\rho_1 \rightarrow \rho_2 \mid X \mid \top \mid @ \mid \$$
Types A	::=	$\mathsf{slide} \mid \bullet \mid \ldots$
Kinds K	::=	$\mathrm{type} \mid \ldots$
Hyperkinds H	::=	$\mathrm{kind} \mid \ldots$
Sorts σ	::=	$a \mid k \mid h \mid \square \mid \bigcirc \mid * \mid \star$
Universes L	::=	$A \mid K \mid H \mid \ldots$
Cosmologies $\mathfrak{C}$	::=	$\ldots$
Philosophies $\mathfrak{P}$	::=	$\ldots$
WTFs $[\![\overline{\mathfrak{M}}^{\circ}]\!]$	::=	$[\![\overline{\mathfrak{M}}^{\circ}]\!] \mid [\![\overline{\mathfrak{M}}^{\circ}]\!]$

Figure 1: Pile of Money

2.1 Slides

Given the uniformity of presentation styles, we can easily classify slides by their type.

Slide	Meaning
intro	Introduction Slide
outline	Outline Slide
diag	Intimidating Diagram
cont($\vec{\beta}$)	Slide With Actual Content
conc	Conclusion Slide

2.2 Bullet Points

We similarly divide bullet points by their function.

Bullet	Meaning
π	Plain Prose
ϵ	Excuses
μ	Misdirection
ϕ	Formula
ι	Inference Rule
ν	Non Sequitur
α	Impenetr. Mass of Abbrevs.
γ	Impenetrable Mass of Greek
$\lambda(\vec{\beta})$	Nested Bullet List

2.3 Typing

Nobody really looks at most of the obvious, run-of-the-mill, completely standard judgments and inference rules such as

$$\Gamma \vdash S : \mathsf{slide}$$

or

$$\frac{\Gamma \vdash \beta_i : \bullet \qquad (\forall i)}{\Gamma \vdash \mathsf{intro} + \mathsf{outline} + \mathsf{diag} + \mathsf{cont}(\vec{\beta}) + \mathsf{conc} : \mathsf{presentation}}$$

anyway, so we decided not to waste space on any more of them except for the most alarming ones. On the other hand, the remaining inference rules have been determined to be unsuitable for human consumption, and we prudently omit them also. We expect them to be used in animal feed, and common garden mulch.

2.4 Refinements

The essential aim of our proposal is to enable the description of refinements of the type of presentations. A presentation that contains too few graphs, too many or too few inference rules, lacks a cute animation at the end, or completely fails to explain how to achieve the research result claimed and uses feeble misdirection to avoid drawing attention to this fact should be rejected.

2.5 Aspects

On the other hand, aspects, as any OOP researcher worth his or her salt knows, are a feature of verbs use to indicate how the action of a verb takes place over time. For example, there are the following aspects.

Aspect	Meaning
Imperfect	Action still going on
Perfect	Action completed
Pluperfect	Sounds funny
Aorist	Sounds even funnier
Simple(r) Past	Hmph! Kids these days!
Progressive	Drives Prius, Eats Granola
"Around" Advice	For debugging
"Before" Advice	For logging
"After" Advice	Too late, therefore useless

Our contribution to Aspect-Oriented Presentation is the **powerpoint-cut**, by which a slide can be inserted at the last minute just as question is asked. This works by *[XXX remember to write this section before submitting!]*.

3 Related Work

While no previous work has presented an exact and complete formal definition of a type system for presentations there has been occasional interest in the intersection between computer science and theatrical presentation construed more broadly. Most of this work has focused on practical systems that are immediately deployable in the field. For example, Hopper, Murger, and Snivelin invented the so-called 'HMS Semaphore' [HMS01] to prevent audience members from beginning concurrent computations ('impertinent questions') and thereby preventing the talk from achieving hard real-time bounds. The Myfair Scheduler [HH64] attempts to provide the same guarantees by refining the language.

More type-theoretic approaches to the problem include the invention of the *phantom type of the operad* due to by Andrew Lloyd Weppél. [Wep92]

Håvard Bringinda and his students at the University of Oslo have been working trying to unify statistical mechanics and category theory. We hope the study of Bringinda Noise, and Bringinda Functors to be fruitful.

4 Future Work

We hope to extend the current type system to include other expositional components of computer science research, in particular conference and journal papers, and technical reports. This should be quite reasonable if we use for document preparation a modern functional programming language. The only obstacle to ensuring of such documents that they contain no errors or other anomalies is `*** ERROR: Redundant Hypokind in SECTION $0xFE 'future work' at ptsptspts.mtx`

5 Conclusion

References

[HH64] A. Hepburn and R Harrison. The Myfair scheduler. In J. Halpern, editor, *Proceedins of th' Haighth Annual Symposium on the Linguistics what is used in Computer Science (LICS'01)*, pages 221–230, Boston, Massachusetts, May 1964. IEEE Computer Society Press.

[HMS01] B. Hopper, B. Murger, and B. Snivelin. A concrete proposal for eliminating interruptions. Technical report, Adobe Systems, 2001.

[Wep92] Andrew Lloyd Weppél. Phantom type of the operad. Unpublished manuscript, 1992.

All Your Trace Are Belong to Us*

Daniel K. Lee

Carnegie Mellon University

1 In Ph.D. 2007

1.1 Code was beginning

```
Captain: What happen ?
Mechanic: Somebody deref us the NULL.
Operator: We get stderr.
Captain: What !
Operator: Bug screen turn on.
Captain: It's you !!
Cats: How are you gentlemen !!
Cats: All your trace are belong to us.
Cats: You are on the fault to segmentation.
Captain: What you say !!
Cats: You have no chance to revert make your commit.
Cats: Ha ha ha ha ....
Operator: Captain !!
Captain: Take off every 'star'!!
Captain: You know what you doing.
Captain: Move 'Star'.
Captain: For great safety.
```

*This work is partially supported by the National Science Foundation under a Graduate Research Fellowship and D's Six Pax and Dogz.

Fried Chicken Bucket Processes

Mary McGlohon
Machine Learning Department
Carnegie Mellon University
mcglohon@cmu.edu

Keywords: Stochastic processes, graphical models, hierarchical models

Abstract

Chinese restaurant processes are useful hierarchical models; however, they make certain assumptions on finiteness that may not be appropriate for modeling some phenomena. Therefore, we introduce fried chicken bucket processes (FCBP) that involve different sampling methods. We also introduce spork notation as a simple way of representing this model.

1 Introduction

Chinese restaurant processes and Indian buffet processes are useful in a number of domains for statistical modeling. This work introduces a new model, the Fried chicken bucket process, and presents spork notation, a useful representation for FCBP's and other graphical models.

To the author's knowledge this is the first restaurant-related model that samples from continuous distributions in addition to discrete ones.

2 Related work

It is useful to describe work that has inspired this paper.

2.1 Chinese restaurant processes

The Chinese restaurant process (CRP) is a stochastic process that produces a distribution on partitions of integers [1]. To visualize, one imagines a Chinese restaurant with an infinite number of tables. Customers arrive one at a time. As each arrives, he decides which table to sit at based on the following distribution similar to a Dirichlet distribution:

$$p(Table_i|n) = \frac{n(i)}{\gamma + n - 1}$$

$$p(NewTable|n) = \frac{\gamma}{\gamma + n - 1}$$

where n is the number of previous customers and $n(i)$ is the number seated at table i.

This may be extended into hierarchies such as the customers also choosing from an infinite number of Chinese restaurants [2]. This process also inspired Griffiths and Ghahramani to describe the Indian buffet process for infinite latent feature models, as shown in [5] and applied again by Thibaux and Jordan in [6].

2.2 Plate notation

In high dimensional problems, representation comes in the form of very large graphical models with many nodes. Formerly researchers simply had their graduate students draw all the nodes. Then, in 1994, Buntine introduced plate notation [3], which drastically reduced the work required to draw a graphical model, and has made it possible for today's machine learning graduate students to focus their efforts on maintaining statistics-related entries on Wikipedia. The plate notation simply groups together nodes that are duplicated– that is, have the same interior-exterior links. An example, flips of a thumb tack, is shown in Figure 1.

3 Fried Chicken Bucket Processes

3.1 Description of model

On the top level, one imagines a fried chicken restaurant with a chicken generating function (cgf): that is, a distribution of chicken parts from which the

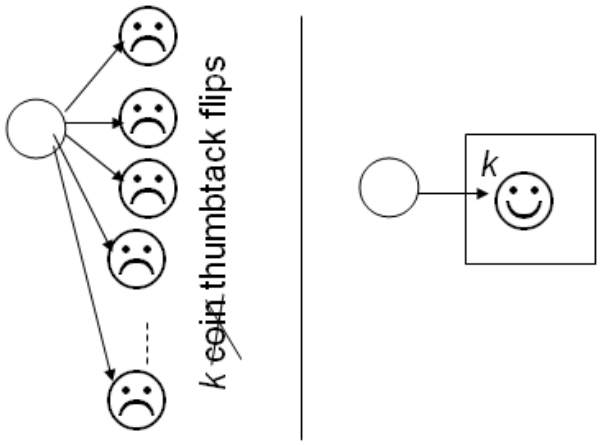


Figure 1: *A graphical model without plate notation (left) and with plate notation (right).*

buckets are made. The restaurant also serves homogeneous okra, coleslaw, or other side dishes which may be treated as continuous.

A family orders a n-piece bucket of fried chicken, which begins the next level. From the cgf, n pieces of fried chicken are drawn, making a much coarser distribution of chicken parts. The family also takes sides. Once the family drives home and spreads dinner on the table, each of k family members chooses chicken pieces from the bucket. Draws are random to avoid squabbles, and the distribution is obviously without replacement[1]. After chicken is drawn, each family member chooses a continuous amount of side dishes. It is well known that the fried chicken runs out while there are often leftover side dishes; therefore for this model we assume that coleslaw and okra are infinite as well as continuous. However, the amount of these dishes may be conditional on the discrete pieces of chicken that were drawn from the bucket, as paper plates have finite capacity.

[1]Sampling with replacement would be unsanitary.

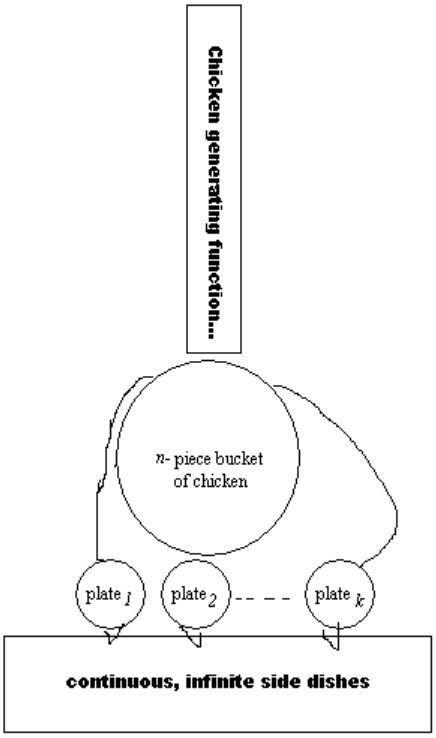


Figure 2: *A FCBP in spork notation.*

3.2 Illustration of model

We can best illustrate the FCBP using a piece of hardware related to the fried chicken bucket: the spork. The cfg (suggested through the handle of the spork) generates the bucket in the reservoir of the spoonlike part. From the bucket, the plates result (prongs), which then "pick" items from the continuous and infinite side dishes. This is shown in Figure 2.

We propose that spork notation be used for any process where a discrete sampling influences a subsequent continuous sampling.

3.3 Instances of model

For theatrical purposes one may choose to specify the cgf. The most obvious choice is a multinomial distribution, with one p_i for each chicken part that may go into the bucket, where $\sum_i p_i = 1$. For example, we might choose $(p_{leg} = .3, p_{breast} = .39, p_{wing} = .3, p_{beak} = .01)$.

4 Applications of FCBP

Many phenomena may be modeled as an interaction between a discrete sampling that influences the way in which a continuous sampling behaves. One may think of mixture models in this fashion; the prongs of the spork may be considered k classes from which different continuous distributions of variables may result. This is significant because mixture models and the methods are sometimes difficult to grasp, and machine learning concepts are easier to understand when they are presented using culinary examples [4].

5 Future Work

It would be desired to extend FCBP's to yet another hierarchy. For instance, one might imagine a strip mall, college campus, or region of a country with an infinite number of fast food stands and allow mixing proportions on a family's dinner table. This, and other further applications of FCBP are left as an exercise to the reader.

6 Conclusion

Machine learning researchers need to stop having meetings when they're hungry.

References

[1] D. J. Aldous. Exchangeability and related topics. *École 'e' té de probabilités de Saint-Flour XIII-1983. Lecture Notes in Mathematics*, 1117.

[2] D. Blei, T. Gri, M. Jordan, and J. Tenenbaum. Hierarchical topic models and the nested chinese restaurant process, 2004.

[3] W. L. Buntine. Operations for learning with graphical models. *Journal of Artificial Intelligence Research*, 2:159–225, 1994.

[4] K. El-Arini. Pizza delivery processes. In *Machine learning office conversations*, 2006.

[5] T. Griffiths and Z. Ghahramani. Infinite latent feature models and the indian buffet process, 2005.

[6] R. Thibaux and M. I. Jordan. Hierarchical beta processes and the indian buffet process. Technical report, University of California, Berkeley.

Manifest Adequacy*

Daniel K. Lee Robert J. Simmons

Carnegie Mellon University

1 First beer

"Wrong? I can't be wrong! I'm never wrong! I refuse to be wrong!"
– Anonymous sober type theorist

Out of a well-founded skepticism for the ability of paper to correct for the errors introduced by smudging and smearing caused by the coffee, beer, and blood spills that are standard hazards in the daily lives of programming language researchers, it has been decided that the future of programming languages research is in the formalization of languages as specifications for computer proof assistants. Commonly, the encoding of the language is as a script written in the programming language used to interact with the proof assistant. In the LF community, a great deal of sober thought has been given to the process in which the LF encoding of a language can be verified to be an *adequate* representation of the intended language. This process is performed by verifying there exists a *compositional bijection* between the LF encoding and the intended language. The failure of an adequacy proof generally implies someone wasted time working on a language he didn't even care about. Because adequacy proofs require time that could be more gainfully spent proving interesting theorems, reading Livejournals, marking the Wean blackboards with type theory, durnk dialing exgirlfriends, grading problem sets, eating tube meat, planning TGs, creating fonts, hunting penguin thieves, or playing internet poker, they are *never* performed in practice.

2 Second beer

We present a novel way of encoding the compositional bijection between the Twelf encodings of a language and the language it is meant to represent. We call this operator I, but in order to retain an air of mystery we will refuse to define the semantics of I here. However, we have a napkin proof that I is reflexive and idempotent. Under the I operator, every Twelf signature is manifestly an adequate representation of some language. Additionally, it is general knowledge that languages encoded in Twelf are superior to those never encoded in Twelf, anyway.

*This work is partially supported by the National Science Foundation under a Graduate Research Fellowship and D's Six Pax and Dogz.

```
tp : type

o : tp.
arrow : tp.

exp : type.

* : exp.
app : exp -> exp -> exp.
lam : tp -> (exp -> exp) -> exp.
```

Figure 1: `stlc.elf`

```
tp : type.

o : tp.
arrow : tp.

exp : type.

* : exp.
app : exp -> exp -> exp.
lam : tp -> (exp -> exp) -> exp.
```

Figure 2: I(`stlc.elf`)

3 Third and subsequent beers

We have discovered an implementation of I in the standard Unix toolset. Assuming the encoded language is in `durnken-logic.elf`, the following command generates the represented language in `durnken-logic-actual.elf`:

```
% cp durnken-logic.elf durnken-logic-actual.elf
```

Additionally, we have discovered a verifier for testing the correctness of an application of I. The successful completion of the following command with no output verifies that `durnken-logic-actual.elf` is I(`durnken-logic.elf`)

```
% diff durnken-logic.elf durnken-logic-actual.elf
```

Anecdotal evidence states that the Unix implementations of I and its verifier are pretty damn fast. The extant pervasiveness of the implementation of the I operator and its verifier on standard computing environments demonstrate their fundamental and foundational importance.

4 Final beer

> *"Oh schnap, I've got it! Without a way to be wrong, I can't help but be right!"*
> – *Anonymous durnken type theorist*

In conclusion, we have suggested a formulation of the only adequacy argument we will ever bother to make.

Methods and Uses of Graph Demoralization

Mary McGlohon
Machine Learning Department
Carnegie Mellon University
mcglohon@cmu.edu

Abstract

Moralizing graphs is useful in understanding independence relations in a directed graph and in converting to an undirected graph. However, the method of demoralizing graphs has not been properly addressed. This work presents graph demoralization. It describes the various methods for demoralizing graphs, and addresses applications of demoralization.

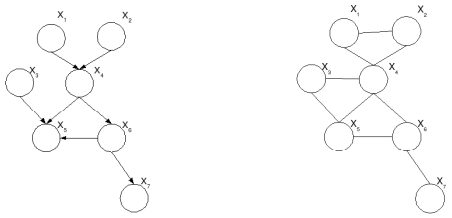
Keywords: Machine Unlearning, Graphic XXX Models!!!, Probabilistic Ethics, Disbelief Propagation, Probably Approximately Researching

1 Introduction

Moralization is an important tool in understanding independence relations in graphical models. However, its dual, demoralization, is an underrepresented concept in the research. This work presents three methods for demoralizing graphs: isolation, misdirection, and disbelief propagation. These methods are based on findings in business management or sociology or some other only vaguely related fields, allowing this work to be submitted to a wider range of journals.

2 Preliminaries

2.1 Probabilistic graphical models. Probabilistic models are commonly illustrated using graphs. In a probabilistic graph, random variables are represented by nodes, and dependencies between random variables are represented by directed edges. The typical parent-child relationships between nodes in graph theory apply, and such relations illustrate the dependencies in the graph. For instance, in Figure 1(a), the value of X_3 is dependent on the value of X_1 and X_2. In a directed graph, only nodes at the receiving end of the edges are dependent on their parents. Parents are independent. More about independencies and such can be found in [6]. For details on inference algorithms, one may consult [2].



(a) Original, immoral graph (b) Moralized graph

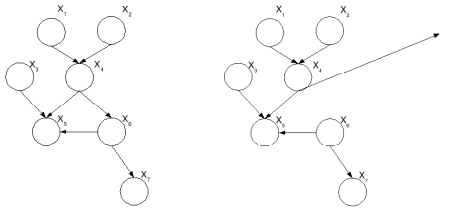
Figure 1: **Moralizing a graph.** In the origial graph (a), there are two immoralities: that of the parents of X_4 and that of the parents of X_6. To moralize the graph you marry the parents and remove directions.

2.2 Moralizing Moralization is a method used for independence relations between directed and undirected graphs. A graph immorality occurs when a child has two or more parents that without dependencies between them. Also note that this isn't one of those marriages you only get for tax breaks; there are real dependencies here.[1] This prevents divorcing nodes, which is an even more immoral matter in the language of probabilistic graphical models than illegitimate children. Once parents are married all other directions in the graph must be removed; otherwise you get a hybrid graph of sorts called a PDAG, very thoroughly described in [3].

3 Methods of demoralization

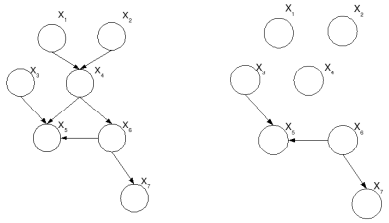
3.1 Misdirecting. Research in social group theory has indicated that groups of agents often become demoralized when they are misdirected [5]. We apply this idea to demoralizing graphs. This is shown in Figure 2, where the edge from node X_4 to X_6 is removed and instead misdirected off the edge of the page. This would cause great confusion in the probability distribution and independence relations in G, thus demoralizing

[1] Technically, the parent nodes may still be independent but they act like they're dependent. This commonly occurs in marriages, although not as often as the dual case.



(a) Original graph (b) Misdirected graph

Figure 2: **Graph demoralization through misdirection.** The original graph (a) is modified by removing the edge from X_4 to X_6 and redirecting it off the edge of the figure and into the middle of nowhere(b).



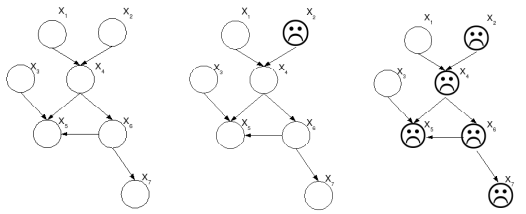
(a) Original graph (b) Isolated graph

Figure 3: **Graph demoralization through isolating.** The original graph (a) is modified by removing all edges between X_4 and other nodes, severing dependencies in the graph and making inference probably approximately nearly impossible(b).

the graph.

3.2 Isolation. Social group theory research also indicates that dysfunction by demoralization occurs when members of the group are isolated [5]. This has a natural application to graphs. We simply pick some nodes to isolate, removing coherence and severing dependencies in the graph. This has the potential for great demoralization. This is shown in Figure 3, where node X_4 is isolated. Its ties to all other nodes are severed, making the previously coherent graph into 4 separate graphs. Inference is nearly impossible in such a system. If you were a graph, you'd be pretty damn demoralized at that point.

3.3 Disbelief conditioning and propagation. The third method for demoralizing graphs is to modify one or more nodes in the graph. We begin by conditioning disbelief upon one or more nodes. This method of demoralization may be more efficient if we also propagate this belief conditioning through the graph. This



(a) Original (b) Modified (c) Propagated

Figure 4: **Graph demoralization through modification and propagation.** The original graph (a) is modified by one node (b), and the modification is propagated over the directed edges through the graph (c).

process, called disbelief propagation, is shown in Figure 4. We show the original graph, the modified graph, and the propagated graph. After propagation the graph is almost fully demoralized. See how sad it looks?

4 Applications of demoralization and discussion on moral state of statistics and statistical machine learning

Demoralization of graphs would seemingly have no applications other than to sate the sadistic susceptibility of statisticians. Sating sadistic statisticians is, of course, an important application that is consistently in use. For instance, simply the word "statistics" is a cruel word to have in the English language, as it is nearly impossible to pronounce, particularly while inebriated– further cruelty lies in that statistics problem sets are most tolerable while in such a state[2]. Another example of sadism in statistics is the long-running turf war between Bayesians and frequentists, that has led to such atrocities as bicycle drive-bys and the bloody showdown that was the rap battle between Emcee MC and The Unbiased M.L.E [1].

Another use of demoralization is application to privacy and security: to prevent inferences. Suppose one had a graphical model, but wanted to keep it secret. For instance, suppose one was embarrassed at his shabby tennis serve and did not want other people to know his $P(PlayTennis|Rain = False)$ (a case discussed in depth in [4]. One could demoralize the graph and prevent others from making inferences on the probability distribution.

[2]Not completed well, mind you, but what sort of expectations can there be when it's a field that accepts 95% confidence as 100%.

5 Conclusion

This work has demonstrated three methods for demoralization of graphs: misdirecting, isolating, and disbelief propagation. Demoralization is useful for adding to the social references in probabilistic graphical models (among ranks with faithfulness, morality, and swinging couples [3, 7]), for providing statistical machine learning folk amusement, and for applications in data privacy.

Acknowledgements

This material is most likely supported by a generous grant the Probably Approximately Foundation. The author was partially supported by a National Science Foundation Graduate Research Fellowship and pizza from Pile of Money grant no. WH-4625.

References

[1] A. Arnold. *Chronicles of the Bayesian-Frequentist Wars*. somewhere in Europe with .75 probability, 1999.

[2] C. Bishop. *Pattern Recognition and Machine Learning: 23 cents cheaper per page than Tom Mitchell's book*. Springer Texts, New York, 2006.

[3] D. Koller and N. Friedman. *Probabilistic Graphical Models (DRAFT)*. Palo Alto, CA, 2007.

[4] T. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[5] E. Stiehl. Misdirected and isolating groups and their subsequent demoralization. *Conversations with resident business grad student at Machine Learning Department holiday parties*, 2006.

[6] L. Wasserman. *All of Statistics*. Pink Book Publishing, New York.

[7] L. Wasserman and J. Lafferty. *All of Statistical Machine Learning*. Pink Book Publishing, New York.

Finding Nonsense in a Nineteenth Century Logic (Abstract)

Matthew Kehrt
University of Washington
mkehrt@cs.washington.edu

Abstract

It is a truth universally acknowledged that Dodgson's *The Jabberwocky* is a triumph of nineteenth century logic. However, exactly how this poem is to be mapped onto a useful or even a useless logical system has been a matter of some debate. Alonzo Church is known to have gone mad studying the more technical aspects of the poem, resulting in the hideously baroque λ-calculus and its barrage of modern descendants.

We present a novel reinterpretation of the work, not as a brilliant anticipation of Gentzen's natural deduction as it is traditionally seen, but instead as a piece of utter nonsense. Applications to category theory are discussed.

Within the context of this new interpretation, we then turn our sights to its applications in modern computer systems. We present a new programming paradigm, *Programming as Nonsense*, and sketch a language called BRILLIG which allows nonsense expressions to be inserted into program code arbitararily, vastly inhibiting program understanding. The type system for this language is shown to be unsound in deeply unsettling ways. Finally, we connect this to the work of GERALD GAZDAR, because we like his name.

A New Historical Analysis of /ʌ/

Greg Hanneman
Language Enthusiast and Errant Scribbler
Carnegie Mellon University

Abstract

We introduce a novel analysis of the phoneme /ʌ/, tracing its use from its first appearance in ancient speech to its distribution in modern utterances. We analyze its contemporary distribution with a probabilistic model and provide a number of illustrative examples. We also practice writing in the plural even though there is only one of us.

1 Introduction

Experts generally agree that the first language in the world was the one developed by the prehistoric people of what is now southern France [6], expressed among the famous cave paintings discovered at Lascaux in 1940. The more commonly studied depictions involve hunting and gathering scenes, village feasts, and wars against neighboring tribes. Also among them, however, way back on that one reddish rock just to the left of the *bête noire*, is a fragment of writing that puzzled linguists for decades (Figure 1). It was later analyzed and discovered to be the Diamond Jubilee version of Gregg shorthand [2].

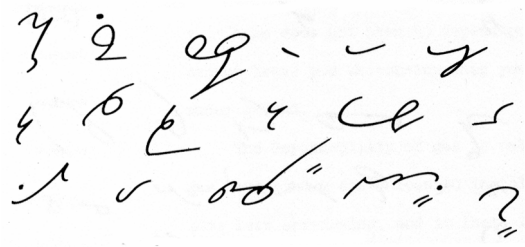


Figure 1: Mysterious fragment of prehistoric writing discovered in a cave at Lascaux.

Of the many phonemes represented in the Lascaux text, one of the most prevalent and useful is /ʌ/. Indeed, the /ʌ/ phoneme's importance, dating from its appearance in well-known prehistoric words like /ʌg/ and continuing to the present day, has been unquestioned for thousands of years. In this paper, we present a novel analysis of /ʌ/, taking especial care to place the use of the sound in its historical and modern contexts. We also describe a series of elicitation experiments designed to provide data on the current use of /ʌ/ in informal speech.

2 Previous Work

Hanneman [3] provided a first-rate introduction of the topic in his very readable recent paper. We cannot hope to improve upon his fine analysis, choosing instead to refer the interested reader to his original work for full details.

3 Experimental Results

The value of /ʌ/ as a space-filler in hesitant and confused speech is well known. Our experiments show that the duration of the sound generally ranges from 550 ms to 1200 ms, with values reaching more than 3000 ms in some speakers. Other so-called "gap" phonemes, such as /a/ or /œ/, are generally less fun to study, so we have no data on them.

3.1 Modern Speech

We do, however, enjoy drawing syntax trees. To handle the appearence of /ʌ/ in a speech stream, we adopt the Penn Treebank annotation UH [4] in each of our examples below. Data taken from two native English speakers between the ages of 23 and 24[1] indicate a somewhat systematic use of UH in casual

[1]The relatively small sample size and homogeneity of our test subjects is due to the fact that they were the only grad students in the lab the night before the SIGBOVIK paper deadline.

speech. A spurious /ʌ/ is most frequenty inserted before a major syntactic constituent (Figure 3.1),

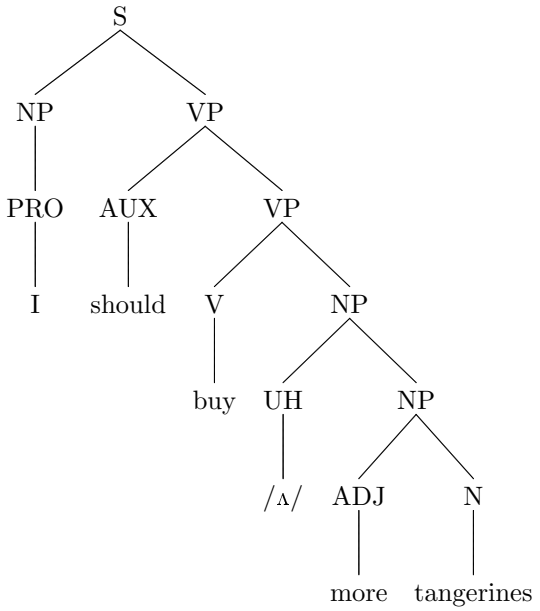


Figure 2: In this example from our corpus, a spurious /ʌ/ occurs before the noun phrase *more tangerines*.

We represent this pre-constituent distribution with a probability model. We explicity model a context-sensitive series of features over lexicalized constituents. In this case, we let c be the syntactic category of the constituent (NP, VP, etc.), h be the lexicalized head word of the constituent (*tangerines*, etc.), and p the current phase of the moon expressed as a real number on the range $[-1, 1]$. We also keep track of a base probability, u, of a /ʌ/ insertion for each speaker s. This gives us the following model:

$$P(u, c, h, p, s) = P(c, h, p)P(u \mid s) \tag{1}$$

The title of this paper, however, promised a *historical* analysis of /ʌ/. Therefore we conclude our discussion of current uses and move on to the next subsection.

3.2 Historical Speech

The earliest wax phonograph cylinders now extant provide a fascinating look at the speech of prehistoric man. (See, for example, Sage [5].) From the beginning, the prevalence of /ʌ/ in some utterances, such as /ʌgə.wʌgə.gʌ/, can approach the level of the letter *e* in modern English texts. These examples show the pervasive extent of the phoneme in ancient speech, and also indicate how a modern player might win at IPA hangman against a Cro-Magnon opponent.

In more modern times, /ʌ/ has continued to be used upon important occasions of all kinds, ranging from Julius Caesar's famous dying words "*Et tu, Brute? /ʌ/!*" in 44 B.C. to Queen Victoria's oft-quoted "/ʌ/ , we are not amused." Modern writers, though, who have creative differences with the International Phonetic Alphabet generally prefer to render /ʌ/ as "uh" in normal text; modern copy editors, who insist on regularity of punctuation, set it off with commas. In this form, the sound has shown up in various contemporary contexts, such as "Cake or, uh, death?" and "This is, uh, Spartaaaaa!" The authors feel that further explication beyond these perceptive examples will not be necessary.

4 Error Analysis

Propagation of error or uncertainty can be calculated in two distinct ways [1]. The first, known as the *derivative method*, computes the error in function f due to uncertainty in variable x as:

$$\delta_{fx} = \left|\frac{\partial f}{\partial x}\right| \delta_x \tag{2}$$

Once the error due to each variable in a function is calculated in this way, they are combined via the quadrature method. For a function $f(a, b, ...n)$:

$$\delta_f = \sqrt{\delta_{fa}^2 + \delta_{fb}^2 + \cdots + \delta_{fn}^2} \tag{3}$$

Error can also be calculated using the so-called *computational method*, which is often preferable to first-year physics students because it requires no calculus. Given a function $f(a, b, ..., n)$, the error due to each variable is simply:

$$\delta_{fx} = |f(a, b, ..., x + \delta_x, ..., n) - f(a, b, ..., x, ..., n)| \tag{4}$$

Then the quadrature method of Equation 3 can be applied.

The application of the above formulas to the data is left as an exercise to the reader [10 points]. Write that down in your copybook now.

References

[1] Department of Physics, Case Western Reserve University. *Lab Manual*, 2006.

[2] John Robert Gregg, Louis Leslie, and Charles Zoubek. *Gregg Shorthand*. McGraw-Hill, 1971.

[3] Greg Hanneman. A new historical analysis of /ʌ/. In *Proceedings of SIGBOVIK: Workshop about Symposium on Robotic Dance Party of*, Pittsburgh, PA, April 2007. Association for Computational Heresy.

[4] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19, 1993.

[5] Glenn Sage. Cylinder of the month. Online, December 1999. http://www.tinfoil.com/cm-9912.htm#e10000.

[6] Ross Steele, Susan St. Onge, and Ronald St. Onge. *La civilisation française en évolution*. Thomson Heinle, 1996.

Wikiplia:
The Free Programming Language That Anyone Can Edit

Tom Murphy VII

1 April 2007

Abstract

We present a new programming language called Wikiplia. The language has an unprecedented level of integration: The system is its own compiler, language definition, documentation, development environment, distributed filesystem, database, revision control system, bootstrapping software license, community message board, and World Wide Web home site. Wikiplia is designed to be Free to a greater extent and in more dimensions than existing languages.

Keywords: Freedom, programming language, software license, wiki, XML, weatherproof footwear and their fastening mechanisms, hyper-driven devices

1 Introduction

One of the most cherished social principles of mankind is freedom,[1] in its many incarnations. More recently, freedom has become an important principle in computer science as well, with the introduction of Free Software licenses such as the GNU GPL,[2] extensible markup languages such as XML,[3] the ability explicitly deallocate memory with the `free(3)` library call, and the widespread availability of Free Herbal V1agara on the World Wide Web.[4]

The aim of this project is to develop a programming language that is as free as possible. We begin by enumerating freedoms that we desire to support. Because freedom is a possession of inestimable value[5] we do not attempt to rank these freedoms; instead, each freedom is "numbered" using a symbol drawn from incomparable sets of glyphs.

Freedom ©: The freedom to tinker. Users should be able to study a program to see how it works, and to make modifications to suit his or her needs. For most software, this means that the programmer needs access to the software's documentation, source code, and UML[6] use case diagrams. This is traditionally achieved through licenses such as the GPL; however, as we will discuss in Section 2 there are special considerations for bootstrapping compilers that render the GPL inadequate for this purpose.

Freedom ℏ: Freedom of expression. Programmers should be able to write their programs using any expressions that they like. Specifically, there should be no prior establishment of arbitrary categories of expression that are excluded, such as those that discriminate on the basis of class, mathematical philosophy, or type.

Freedom ♀: Free to be You and Me. The development of a programming language should not be confined to the bearded academic elite, gazing down upon the programmer fiefs from their stratospheric ivory towers. Wikiplia is the free programming language that *anyone* can edit: from bearded academic elite[7] to congressional staffers[8] to nameless Slashdot[9] trolls. Wikiplia's WWW-based approach means that

[1] Wikipedia, the free encyclopedia: *Freedom*; 2007

[2] Wikipedia, the free encyclopedia: *GNU General Public License*; 2007

[3] Wikipedia, the free encyclopedia: *XML*; 2007

[4] Wikipedia, the free encyclopedia: *Sildenafil*; 2007

[5] Wikipedia, the free encyclopedia: *Cicero*; 2007

[6] Wikipedia, the free encyclopedia: *Unified Modeling Language*; 2007

[7] Wikipedia, the free encyclopedia: *Tenure*; 2007

[8] Wikipedia, the free encyclopedia: *Congressional staffer edits to Wikipedia*

[9] Wikipedia, the free encyclopedia: *Slashdot*; 2007

an Internet connection and compatible WWW hypertext browser is all that's needed to begin on the life-changing journey of programming language design.

Freedom △: Freedom of beer. Users should be able to write software without paying money to a licensing authority or certification program.

Freedom ¶: Freedom to redefine freedom. Freedom should be free, so the definition of freedom should be free to change as the meaning of freedom changes. Wikiplia's license allows for Wikiplia to be distributed in a way that monotonically increases freedom as new concepts and catchphrases of freedom are invented.

Freedom $\sqrt[x^{1.2}]{\widehat{\mathrm{oo}} + \frac{x}{z^2}}$: Freedom of USA #1. Wikiplia is 100% made in the USA and only available in English.[10]

2 Reflections on strapping straps and booting boots

The hallmark of Free software is the GNU General Public License. It is a hereditary license that requires that (a) source code be distributed with the program and (b) modified versions of the program also be licensed under the GPL. The intention is that anyone receiving the software can exercise Freedom © by understanding the source code and modifying it to suit his needs. Clearly any source code will not do: an obfuscated[11] version of the source code cannot be easily understood or modified, even though it is technically "source code." The GPL therefore legally defines source code as the "preferred form" for "making modifications."

Even source code in the preferred form might not be enough to achieve Freedom ©, however. For instance, the program might be written in a mysterious programming language that only the author understands, and that programming language might only be implemented in a private compiler on the author's hard disk.[12] It is therefore reasonable to construe the "preferred form" of the original software to include the implementation of the programming language that the software is written in. Because the programmer might need to fix bugs or extend the programming language implementation in order to modify the original program, he also needs the source code for that language as well. This code must also be written in some language, so the process continues. It can end when one of the programming languages is generally well known enough that there are no practical barriers to understanding it or finding an implementation (examples would include C[13] and ALGOL 58[14]), or so simple that the implementation is essentially non-existent (*e.g.* an assembler implemented directly in machine code).

Another way for this process to terminate is for a programming language to be implemented in itself. This is known as a "bootstrapping compiler." A natural social tendency causes this to be very common: language implementors are more likely to enjoy the language they are implementing, and therefore more likely to choose it to implement the language. But when this process terminates this way, the reader might be left with a suspicious sense that nothing has actually been achieved.[15] Specifically: What freedom-fulfilling use is the source code to an implementation of a mysterious programming language, if that source code is itself in the same mysterious programming language?

Let us concentrate on a more concrete example. The GNU C compiler[16] (licensed under the GPL) is an implementation of the C language with some extensions specific to the compiler. The GCC source code uses some of these extensions. Can the GCC be Free software if it requires the GCC to build? In the extreme case, what if someone were to add an extension to the GCC to enable a new C keyword—called `compile_a_program`—and then replace the entire source code with:

```
int main (int argc, char ** argv) {
  compile_a_program;
  return 0;
}
```

Such code is clearly worthless. Not all subversions of the source code via language extension may be so overt, but we claim that they nonetheless pose a substantial threat to freedom.

We do not wish to limit the programmer's ability to make extensions to a language, since this would

[10] Wikipedia, the free encyclopedia: *Freedom fries*; 2007
[11] Wikipedia, the free encyclopedia: *Obfuscated code*; 2007
[12] Wikipedia, the free encyclopedia: *Hard disk*; 2007
[13] Wikipedia, the free encyclopedia: *C (programming language)*; 2007
[14] Wikipedia, the free encyclopedia: *ALGOL 58*; 2007
[15] In the case of a LOGO interpreter implemented in LOGO, we could say that this is then "turtles all the way down."
[16] Wikipedia, the free encyclopedia: *GNU Compiler Collection*; 2007

also toe-step Freedom ©. We then conclude that the licensing terms must be expanded in order to provide more than "source code." We propose that not only the source, but the source code's *history*, must be made available.

2.1 Revision control

Computer scientists use revision control[17] to track changes to software and to coordinate development between multiple programmers. This has been true for thousands of years. Popular revision control systems such as CVS[18] and Subversion[19] allow for code to be concurrently modified by two or more developers and then have their changes integrated after the fact by an explicit "check in" and conflict resolution phase.

It may naïvely seem that publishing the entire CVS history of a project would solve the issue with language extensions: By inspecting the revision that introduced the `compile_a_program` feature (but prior to the replacement of the GCC with the minuscule version above), one could then see its implementation and then know what it means. For certain patterns of development this does indeed suffice. However, programmers are not forced to check in their changes except at their own whims, as determined by social conventions; a programmer might make the private addition of the keyword `compile_a_program`, then rewrite the GCC to use it, and only then check in this change as one revision. For this action he will surely be rebuked by his fellow programmers; none of the other developers can compile the new version of the code without access to the intermediate revision! This social pressure would also naïvely seem to be enough to address the problem, but more insidious scenarios yet obtain.

As a concrete example, suppose there are two programmers called K and R. Each is modifying the GCC with the purpose of adding a new character constant, `'\c'`. K and R start at revision 100 of the GCC. K finds the case analysis for parsing character constants:

```
/* REVISION 100 (K) */
switch(ch) {
  case 'n': return '\n';
  case 'r': return '\r';
  ⋮
  default: abort("bad char constant");
}
```

He adds a case for his extension, without using the extension, and checks this in as revision 101.

```
/* REVISION 101 (K) */
switch(ch) {
  case 'n': return '\n';
  case 'r': return '\r';
  ⋮
  case 'c': return 257;
  default: abort("bad char constant");
}
```

Meanwhile, R has similar (but not identical) inspiration and modifies his copy of the compiler:

```
/* REVISION 100 (R) */
switch(ch) {
  case 'n': return '\n';
  case 'r': return '\r';
  ⋮
  case 'c': return 8675309;
  default: abort("bad char constant");
}
```

He does not commit his code because he is wary of the time-consuming conflict resolution phase and is late for a date with K's estranged wife who is fed up with K's all-night hacking binges. He burns rubber in his 2007 Honda Civic[20] with aftermarket spoiler for a night on the town, believing that a healthy well-rounded programmer spends more or less equal nights basking in the pale amber glow of the teletype as waking up with a few missing teeth naked and norovirused in some midtown alleyway with his wallet barely out of reach but empty anyway, having amply exercised Freedom △.

Meanwhile, K continues extending the GCC, using the extension to implement itself. He checks in this code with no conflicts:

[17] Wikipedia, the free encyclopedia: *Revision control*; 2007
[18] Wikipedia, the free encyclopedia: *Concurrent Versions System*; 2007
[19] Wikipedia, the free encyclopedia: *Subversion (software)*; 2007
[20] Wikipedia, the free encyclopedia: *Honda Civic*; 2007

```
/* REVISION 102 (K) */
switch(ch) {
  case 'n': return '\n';
  case 'r': return '\r';
  ⋮
  case 'c': return '\c';
  default: abort("bad char constant");
}
```

K punches out at `1130 UTC`;[21] just as R returns from his adventure. R's confidence bolstered, he finishes his extension effort, following best practices and implementing the extension using itself:

```
/* REVISION 100 (R) */
switch(ch) {
  case 'n': return '\n';
  case 'r': return '\r';
  ⋮
  case 'c': return '\c';
  default: abort("bad char constant");
}
```

He now decides to commit his changes (forgetting that he did not commit the intermediate revision). To do so he updates to the newest revision, 102, and sees that there are no conflicts—in fact, revision 102 already contains his changes! Believing that his changes are therefore compatible, he continues hacking.

After this scenario, K and R believe they are working on the same programming language—after all, it has the same source code—but their minor bifurcation in development history means that they have forever different meanings of the `'\c'` extension. This mistake is likely to go unnoticed for some time, and until it is resolved, the meaning of the `'\c'` extension is firmly enslaved in the bipartite penitentiary of double entendre, yearning to be free . . .[22]

[21] Wikipedia, the free encyclopedia: *Coordinated Universal Time*; 2007

[22] Wikipedia, the free encyclopedia: *Information wants to be free*; 2007

2.2 Solution

Based on these scenarios we conclude that extant social measures such as revision control conventions are not enough to guarantee freedom in all circumstances. Even if we think these situations are implausible in the hands of well-intentioned, well-mannered and capable[23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45] software engineers, we wish for our software to remain free even when in the hands of nefarious and crafty factions who would seek to fracture[46] our free software community. We therefore need a technological and legal solution that forces the entire development history to be available. Keeping with Freedom $\sqrt[x^{1.2}]{\widehat{\mathrm{oo}} + \frac{x}{z^2}}$, we call this technology and its license *Total Information Awareness* after the successful project of the US Government with the same name.[47]

Technologically, we develop our system around a primitive notion of revision control in which every change to the system is recorded. Because the system has an integrated editor, every action of a programmer is logged and preserved indefinitely with no extra action necessary on the programmer's part. Such commits are globally atomic, using a single universal repository. (This means that in the above scenario, R would not have been able to forget to commit his intermediate change, and would have been forced to observe his conflict with K.) To protect against the possibility that divergent development paths lead to incompatible compilers, we require that there is only one compiler for any given programming language, which itself exists in the revision control system. Therefore, it is always clear which version of

[23] Wikipedia, the free encyclopedia: *Year 2000 problem*; 2007

[24] Wikipedia, the free encyclopedia: *Northeast Blackout of 2003*

[25] Wikipedia, the free encyclopedia: *Ariane 5 Flight 501*; 2007

[26] Wikipedia, the free encyclopedia: *Mars Climate Orbiter*; 2007

[27] Wikipedia, the free encyclopedia: *Mars Polar Lander*; 2007

[28] Wikipedia, the free encyclopedia: *Mars Rover*; 2007

[29] Wikipedia, the free encyclopedia: *Mars Pathfinder*; 2007

[30] Wikipedia, the free encyclopedia: *Gripen#Crashes*; 2007

[31] Wikipedia, the free encyclopedia: *missingno.*; 2007

[32] Wikipedia, the free encyclopedia: *XSS*; 2007

[33] Wikipedia, the free encyclopedia: *Buffer overflow*; 2007

[34] Wikipedia, the free encyclopedia: *Therac-25*; 2007

[35] Wikipedia, the free encyclopedia: *Lothar (storm)*; 2007

[36] Wikipedia, the free encyclopedia: *Mariner 1*; 2007

[37] Wikipedia, the free encyclopedia: *Code Red worm*; 2007

[38] Wikipedia, the free encyclopedia: *SQLSlammer*; 2007

[39] Wikipedia, the free encyclopedia: *Sandstorm (vehicle)*; 2007

[40] Wikipedia, the free encyclopedia: *Samy (XSS)*; 2007

[41] Wikipedia, the free encyclopedia: *Pentium FDIV bug*; 2007

[42] Wikipedia, the free encyclopedia: *MIM-103 Patriot*; 2007

[43] Wikipedia, the free encyclopedia: *Windows 95*; 2007

[44] Wikipedia, the free encyclopedia: *Morris (computer worm)*; 2007

[45] Wikipedia, the free encyclopedia: *USS Yorktown (CG-48)*; 2007

[46] Wikipedia, the free encyclopedia: *Fork (software development)*; 2007

[47] Wikipedia, the free encyclopedia: *Information Awareness Office*; 2007

the compiler was used to produce an executable from source. This also makes the K and R scenario impossible: there is only one compiler and so it is impossible for it to differ from itself.

Legally, all of the software is licensed (Section 5) under terms similar to the GNU GPL, but that define the "source code" to include the entire revision history of the system. In order to be compatible with Freedom ¶, we allow the license itself to be edited, but to ensure that no one can remove freedoms already present in the license, the license includes a provision that allows any *prior* version of the license to be used, at the programmer's option.

2.3 Implementation

Wikiplia, the free programming language that anyone can edit, is implemented as a web-site on the Internet at the address

`http://wikiplia.spacebar.org:2222/`

2.4 Roadmap

The remainder of this paper proceeds as follows. We first present in Section 3 the design of the initial Wikiplia system, which is used to bootstrap the rest of Wikiplia. We then discuss the current state of Wikiplia as of revision 532 in Section 4. We explain the freedom-preserving TIA Public License in Section 5. We conclude with a discussion of unrelated work and plans for the future 6.

3 Core calculus

$$\begin{array}{rcl} X & ::= & \texttt{<}tag\texttt{>}\ X_1\ X_2\ \ldots\ X_n\ \texttt{</}tag\texttt{>} \\ & | & \text{string} \end{array}$$

Figure 1: Syntax of XML

Wikiplia is built upon a core calculus of structured data with primitive revision control. Because we wish to support the freedom to tinker, the structured data take the form of XML (the *extensible* markup language; Figure 1). Similar to the W3C's XML Validation,[48] we allow the quality of an XML document to be assessed via a process called *evaluation*, whose output (if any) is itself an XML document. Selected rules for XML evaluation (the dynamic semantic markup) are given in Figures 2 and 3.[49]

Revision control is accessed through the class of imperative `cvs` judgments. We assume a single global repository, which maps *keys* (strings) to lists of *revisions*. A revision is a monotonically increasing and unique *revision number* (integer) paired with an XML document. The judgment `cvs commit` $s\ X = i$ creates a new revision with revision number i and data X and inserts it under the key s. The judgment `cvs checkout` $s = X$ fetches the most recent revision for the key s,[50] the document X (if no such key exists, then the document is invalid). The judgment `cvs checkout -r` $i\ s = X$ does the same, but for the specific revision number i[51] (if no such revision exists, the document is invalid). Finally, `cvs log` $s = \vec{X}$ fetches all of the revision numbers for the key s, as a series of integers $\vec{X}$.

3.1 Syntax

$$\begin{array}{rcl} E & ::= & \texttt{(}\ E_1\ E_2\ \ldots\ E_n\ \texttt{)} \\ & | & \texttt{"string"} \\ & | & n \\ & | & \text{symbol} \\ & | & \texttt{'E} \end{array}$$

$$\begin{array}{rcl} [\![\texttt{(}\ E_1\ \ldots\ E_n\ \texttt{)}]\!] & = & \texttt{<list>}[\![E_1]\!]\ \ldots\ [\![E_n]\!]\texttt{</list>} \\ [\![\texttt{"}s\texttt{"}]\!] & = & \texttt{<string>}s\texttt{</string>} \\ [\![n]\!] & = & \texttt{<int>}n\texttt{</int>} \\ [\![sym]\!] & = & \texttt{<symbol>}sym\texttt{</symbol>} \\ [\![\texttt{'}E]\!] & = & \texttt{<quote>}[\![E]\!]\texttt{</quote>} \end{array}$$

Figure 4: The syntax for the XESP syntax. The recursively defined $[\![\cdot]\!]$ operation converts an XESP expression into an XML document.

XML documents are universally parseable.[52] However, they are difficult to write and read. Therefore, as usual[53] we create a new syntax by which humans can write and read documents and which the computer automatically parses and converts to the eas-

[48] Wikipedia, the free encyclopedia: *XML schema*; 2007

[49] XML documents can be self-correcting through the use of the `handle` primitive, which detects an invalid document and proceeds along an alternative path. We omit the rules for this feature, which requires propagating invalid document status throughout evaluation and thus complicates the rules substantially.

[50] Wikipedia, the free encyclopedia: *Dynamic scope*; 2007

[51] Wikipedia, the free encyclopedia: *Static scope*; 2007

[52] Wikipedia, the free encyclopedia: *Parsing*; 2007

[53] Wikipedia, the free encyclopedia: *RELAX NG*; 2007

$$\frac{}{\Gamma \vdash \mathsf{eval}\ \texttt{<string>}s\texttt{</string>} \mapsto \texttt{<string>}s\texttt{</string>}} \qquad \frac{}{\Gamma \vdash \mathsf{eval}\ \texttt{<quote>}X\texttt{</quote>} \mapsto X}$$

$$\frac{}{\Gamma \vdash \mathsf{eval}\ \texttt{<int>}s\texttt{</int>} \mapsto \texttt{<int>}s\texttt{</int>}} \qquad \frac{}{\Gamma \vdash \mathsf{eval}\ \texttt{<prim>}s\texttt{</prim>} \mapsto \texttt{<prim>}s\texttt{</prim>}}$$

$$\frac{s\ \mathsf{prim}}{\Gamma \vdash \mathsf{eval}\ \texttt{<symbol>}s\texttt{</symbol>} \mapsto \texttt{<prim>}s\texttt{</prim>}} \qquad \frac{\Gamma(s) = X}{\Gamma \vdash \mathsf{eval}\ \texttt{<symbol>}s\texttt{</symbol>} \mapsto X}$$

$$\frac{}{\Gamma \vdash \mathsf{eval}\ \texttt{<closure>}\Gamma\ s\ X\texttt{</closure>} \mapsto \texttt{<closure>}\Gamma\ s\ X\texttt{</closure>}}$$

$$\frac{\Gamma \vdash \mathsf{eval}\ X_1 \mapsto X_1' \quad \cdots \quad \Gamma \vdash \mathsf{eval}\ X_n \mapsto X_n' \quad \Gamma \vdash \mathsf{rate}\ X_1'\ \ldots\ X_n' \mapsto X'}{\Gamma \vdash \mathsf{eval}\ \texttt{<list>}X_1\ \ldots\ X_n\texttt{</list>} \mapsto X'}$$

Figure 2: Evaluation of XML, part 1. The judgment $\Gamma \vdash \mathsf{eval}\ X \mapsto X'$ indicates an assessment of the document X with value X'. The judgment rate is an auxiliary assessment of a list of documents. It is defined in Figure 3. $\vec{X}$ is shorthand for a possibly empty sequence of XML documents. Γ is itself an XML document of the form <list><symbol>s_1</symbol> X_1 ... <symbol>s_n</symbol> X_n</list>. We take the judgment $\Gamma(s) = X$ to produce the leftmost X_i in Γ such that s_i is s. $\Gamma, s = X$ is <list><symbol>s</symbol> $X\vec{X}$</list> if Γ is <list>$\vec{X}$</list>. The judgment s prim holds when s is one of insert, head, read, abort, lambda, list, cons, quote, string, xcase, size, sub, substr, handle, parse, eval, eq, +, -, int, history, let, or if.

ily parseable XML syntax and back to the new syntax, reducing complexity.[54] This compact syntax is based upon parentheses rather than tags: The XML document <list>X_1 X_2</list> is instead written $(X_1\ X_2)$. Note that the closing parenthesis is not *named* as in XML, which makes parsing difficult because the computer must *guess* which parenthesis belongs to which other parenthesis. We therefore call this compact syntax XESP because it basically *reads the programmer's mind*[55] to guess what the name of the closing parenthesis should be. The grammar for XESP is given in Figure 4 along with the translation to XML documents. From now on, we use the XESP syntax in our examples.

3.2 Implementation

Wikiplia is implemented as a World Wide Web HomeSite, which allows for easy access from any location.

The system is implemented as a Standard ML[56] program of approximately 1,000 lines.[57] This program is designed to be minimal: it consists of a web server, a revision control system, and facilities for evaluating XESP documents. It also contains a very minimal bootstrapping "compiler" for XESP documents, with its boots manually strapped. From this tiny core we then develop the remainder of Wikiplia using Wikiplia itself.

Some may balk at the choice of Standard ML, as the language is miserably non-free: First, while many of the major implementations are GNU or BSD-licensed,[58] all are implemented in Standard ML itself, yielding the bootstrapping problem described earlier. Second, the mathematical Definition of Standard ML book is only available as a copyrighted publication of MIT Press,[59] not even *ostensibly* in a free manner. However, the performance considerations of the server and evaluator—and the lack of suitable free alternatives—force us to settle for such subjugation.

3.2.1 Web Server

The web server's job is simple. It runs in a loop, accepting a single connection, setting up an initial environment Γ for evaluation (which associates the symbols request.url, request.ip and request.time with

[54] For efficiency, the Wikiplia implementation is optimized to lazily perform these translations, so that the document is never represented in XML form.

[55] Wikipedia, the free encyclopedia: *Extra-sensory perception*; 2007

[56] Wikipedia, the free encyclopedia: *Standard ML*; 2007

[57] This count does not include general purpose libraries, such as a networking library.

[58] Wikipedia, the free encyclopedia: *BSD licenses*; 2007

[59] Wikipedia, the free encyclopedia: *MIT Press*; 2007

$$\frac{}{\Gamma \vdash \text{rate } \texttt{<prim>list</prim>}\ \vec{X} \mapsto \texttt{<list>}\vec{X}\texttt{</list>}}$$

$$\frac{}{\Gamma \vdash \text{rate } \texttt{<prim>cons</prim>}\ X\ \texttt{<list>}\vec{X}\texttt{</list>} \mapsto \texttt{<list>}X\ \vec{X}\texttt{</list>}}$$

$$\frac{}{\Gamma \vdash \text{rate } \texttt{<prim>lambda</prim> <symbol>}s\texttt{</symbol>}\ X \mapsto \texttt{<closure>}\Gamma\ s\ X\texttt{</closure>}}$$

$$\frac{\Gamma', s = \texttt{<list>}\vec{X}\texttt{</list>} \vdash \text{eval } X \mapsto X'}{\Gamma \vdash \text{rate } \texttt{<closure>}\Gamma'\ s\ X\texttt{</closure>}\ \vec{X} \mapsto X'} \qquad \frac{\Gamma \vdash \text{eval } X \mapsto X'}{\Gamma \vdash \text{rate } \texttt{<prim>eval</prim>}\ X \mapsto X'}$$

$$\frac{\Gamma \vdash \text{eval } X \mapsto X'}{\Gamma \vdash \text{rate } \texttt{<prim>xcase</prim> <list></list>}\ X\ \vec{X} \mapsto X'}$$

$$\frac{\Gamma, s_h = X_h, s_t = X_t \vdash \text{eval } X_b \mapsto X'}{\begin{array}{c}\Gamma \vdash \text{rate } \texttt{<prim>xcase</prim> <list>}X_h\vec{X}_t\texttt{</list>}\ X_0 \\ \texttt{<list><symbol>}s_h\texttt{</symbol> <symbol>}s_t\texttt{</symbol>}\ X_b\texttt{</list>}\ \vec{X} \mapsto X'\end{array}}$$

$$\frac{\Gamma, s_q = X \vdash \text{eval } X_b \mapsto X'}{\begin{array}{c}\Gamma \vdash \text{rate } \texttt{<prim>xcase</prim> <quote>}X\texttt{</quote>}\ X_0\ X_1 \\ \texttt{<list><symbol>}s_q\texttt{</symbol>}\ X_b\texttt{</list>}\ \vec{X} \mapsto X'\end{array}}$$

$$\frac{\Gamma \vdash \text{eval } X \mapsto X'}{\Gamma \vdash \text{rate } \texttt{<prim>xcase</prim> <string>}s\texttt{</string>}\ X_0\ X_1\ X_2\ X\ \vec{X} \mapsto X'}$$

$$\frac{\Gamma \vdash \text{eval } X \mapsto X'}{\Gamma \vdash \text{rate } \texttt{<prim>xcase</prim> <int>}i\texttt{</int>}\ X_0\ X_1\ X_2\ X_3\ X\ \vec{X} \mapsto X'}$$

$$\frac{\Gamma, s_b = \texttt{<string>}s\texttt{</string>} \vdash \text{eval } X_b \mapsto X'}{\begin{array}{c}\Gamma \vdash \text{rate } \texttt{<prim>xcase</prim> <symbol>}s\texttt{</symbol>}\ X_0\ X_1\ X_2\ X_3\ X_4 \\ \texttt{<list><symbol>}s_b\texttt{</symbol>}\ X_b\texttt{</list>}\ \vec{X} \mapsto X'\end{array}}$$

$$\frac{t = \texttt{prim}\ or\ \texttt{closure} \quad \Gamma \vdash \text{eval } X \mapsto X'}{\Gamma \vdash \text{rate } \texttt{<prim>xcase</prim> <t>}\vec{X}_t\texttt{</t>}\ X_0\ X_1\ X_2\ X_3\ X_4\ X_5\ X\vec{X} \mapsto X'}$$

$$\frac{}{\Gamma \vdash \text{rate } \texttt{<prim>quote</prim>}\ X \mapsto \texttt{<list>}X\texttt{</list>}}$$

$$\frac{\texttt{cvs checkout } s = X}{\Gamma \vdash \text{rate } \texttt{<prim>head</prim> <string>}s\texttt{</string>} \mapsto X}$$

$$\frac{\texttt{cvs checkout } -\texttt{r}\ i\ s = X}{\Gamma \vdash \text{rate } \texttt{<prim>read</prim> <string>}s\texttt{</string> <int>}i\texttt{</int>} \mapsto X}$$

$$\frac{\texttt{cvs commit } s\ X = i}{\Gamma \vdash \text{rate } \texttt{<prim>insert</prim> <string>}s\texttt{</string>}X \mapsto \texttt{<int>}i\texttt{</int>}}$$

$$\frac{\texttt{cvs log } s = \vec{X}}{\Gamma \vdash \text{rate } \texttt{<prim>history</prim> <string>}s\texttt{</string>} \mapsto \texttt{<list>}\vec{X}\texttt{</list>}}$$

Figure 3: Evaluation of XML, part 2. The rate judgment assesses a sequence of XML expressions. The rules for rating the primitives parse, string, sub, substr, +, -, int, eq and if are omitted for space.

the appropriate values), and then evaluating the document contained at the head of the key `main` in the repository. The result of that evaluation is sent back to the web browser as a string.[60] The server knows nothing else about how Wikiplia works.

The web server is single-threaded (so each request must finish before the next is handled) because we desire global atomicity (Section 2).

3.2.2 Revision Control

The revision control system stores XESP documents and their history. This is mostly a straightforward implementation of the imperative `cvs` judgments given in Section 3. Unlike typical revision control systems, we need to support very large numbers[61] of revisions with small edits (since every change is saved), so the implementation is engineered to make storing small revisions very cheap. Particularly, revisions are aggressively compressed by only storing the newest revision directly, and then a series of difference "plans" that describe how to get the next older revision from the current one. We compute optimal plans using an efficient minimal edit distance[62] calculation at the token[63] level. As of revision 532, the database is only 250 kilobytes.[64]

3.2.3 Document Evaluation

The Wikiplia implementation also has facilities for parsing and evaluating XESP documents. These are a direct implementation of the evaluation rules in Figures 2 and 3.

3.2.4 Bootstrapping

The key `main` of Wikiplia is responsible for decomposing the URL and acting upon it however is appropriate. The goal of the bootstrapping process is to make Wikiplia self-sufficient[65] in the sense that the language can be edited from the web site implemented by `main`. To do so, the web site needs to be able to present the user with an edit box containing the current source of the `main` key and the ability to save his code into the database, overwriting the `main` key, in order to add functionality.

[60] The HTTP result code may be modified if, for example, the resulting document is a HTTP redirect to another URL.

[61] The implementation also supports revision numbers of arbitrary size.

[62] Wikipedia, the free encyclopedia: *Levenshtein distance*; 2007

[63] Wikipedia, the free encyclopedia: *Token (parser)*; 2007

[64] Wikipedia, the free encyclopedia: *Kilobyte*; 2007

[65] Wikipedia, the free encyclopedia: *Self-sufficiency*; 2007

```
(lambda 's '(parse (xcase s 'no '(h _ h))))
```

Figure 5: The initial bootstrapping compiler.

The initial implementation of `main` is provides for the simple ability to edit, save, and view the current version of keys in the repository. These three actions are encoded as the URLs `/edit/key`, `/save/key`, and `/view/_/key`. The view action is straightforward. The edit action displays an HTML textarea[66] containing the current value of the key and a button that submits the changes to the `save` url. The save action is the most complex. First, the submitted document is saved as the new version of the key. Then, if the key is of the form `base.extension`, the database is checked to see if there is a key called `extension:compile`. If so, the XESP document that is there is applied to the input document (a string) to produce a document that is saved at the key `base`. This allows us to develop languages that are automatically compiled when saved.

This initial functionality is implemented directly in the XESP language, whose extension is `b`; the bootstrapping "compiler" is just the built-in parser (Figure 5).

Arranging that the repository contain the correct keys to make this work is subtle. A small initialization phase sets up:

- `b:compile.b` The source code (a string) in Figure 5
- `b:compile` The parsed document corresponding to the above, such that `b:compile` applied to `b:compile.b` yields `b:compile`
- `main.b` The source code of the original "main" program
- `main` The parsed document corresponding to the above, such that `b:compile` applied to `main.b` yields `main`.

We do this by *anticipating*, during the initialization process, the *meaning* of `b:compile` so that we can perform that action (parsing) on `b:compile.b` to produce `b:compile`. This can only be achieved by fiat and this is the essence of bootstrapping.

After this minimal initialization, we can then exclusively use the web interface to develop and extend Wikiplia.

[66] Wikipedia, the free encyclopedia: *Text box*; 2007

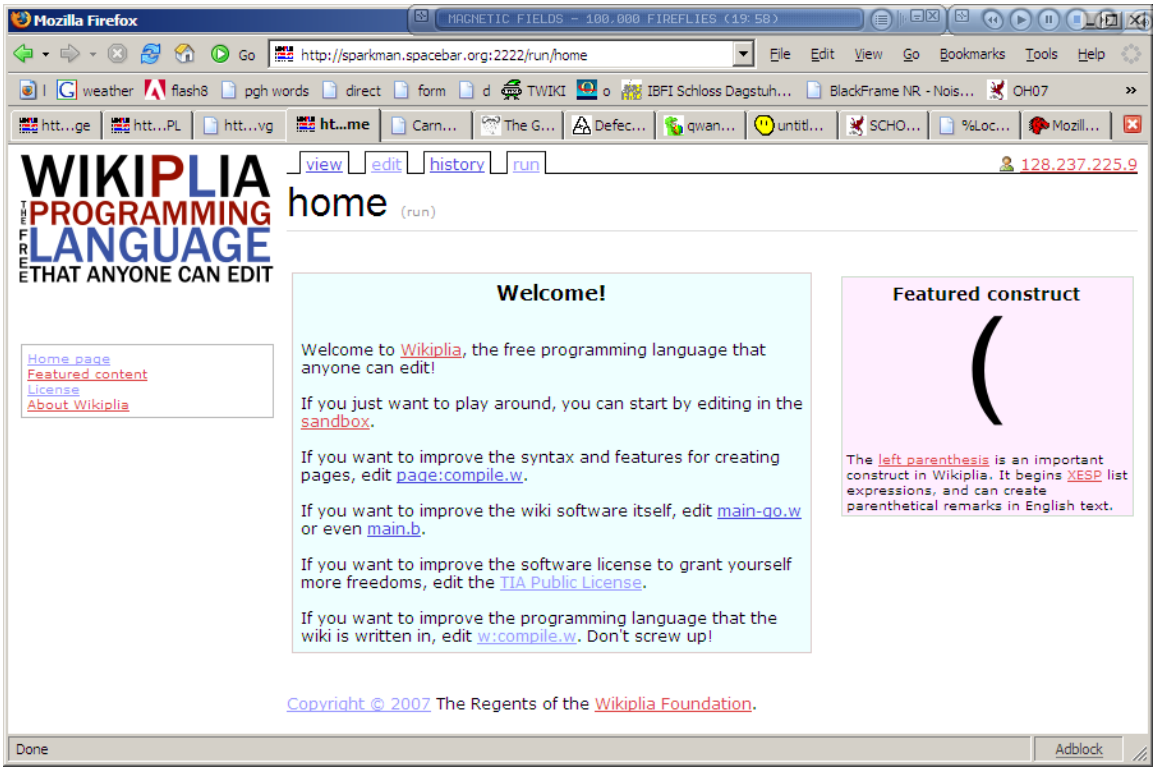


Figure 6: Screenshot of the Wikiplia home page as of revision 532.

4 Revision 532

As of writing, Wikiplia is at revision 532, and has a number of features implemented.

4.1 Interface

The editing interface implemented has been enhanced greatly; Figure 6 shows a screenshot of the main page. The various views of a key are shown with a series of tabs at the top of each page. Each user has a home page named after his IP address,[67] which he can use to catalogue his interests. Various warnings help the user, for example, if he tries to edit a page that was generated by compiling some source code, a warning message suggests that he may wish to edit the source code instead. An ornate logo in SVG[68] adorns the page, and a sidebar provides quick access to the site's features. The logo and graphics for the site are stored in the repository; the new `raw` and `typed` actions allow access to these resources over HTTP so that they may be freely modified.[69] At revision 228 support for metadata was added for each page; a `history` tab now shows the date, revision number, IP address, and edit summary for each change to a key (Figure 7).

It is very easy to make mistakes that render the system unusable. Therefore Wikiplia supports the ability to safely revert to a previous version of a key. When making changes to the `main` key this ability can be accidentally disabled, so the complex functionality of `main` was split off to a new key called `main-go` at revision 9; the `main` key now only dispatches to `main-go` but provides an `emergency-revert` action that automatiacly reverts `main-go` to its previous revision in case it is damaged and the site is unusable.

[67] Wikipedia, the free encyclopedia: *IP address*; 2007

[68] Wikipedia, the free encyclopedia: *Scalable Vector Graphics*; 2007

[69] Because Wikiplia is dogmatically forward-looking, the graphics require SVG support in the browser and support for the `data:` URL format, a combination only found in the newest versions of the Mozilla Firefox. Wikiplia is compatible with incompatible browsers, however, simply displaying a crapified version of each page.

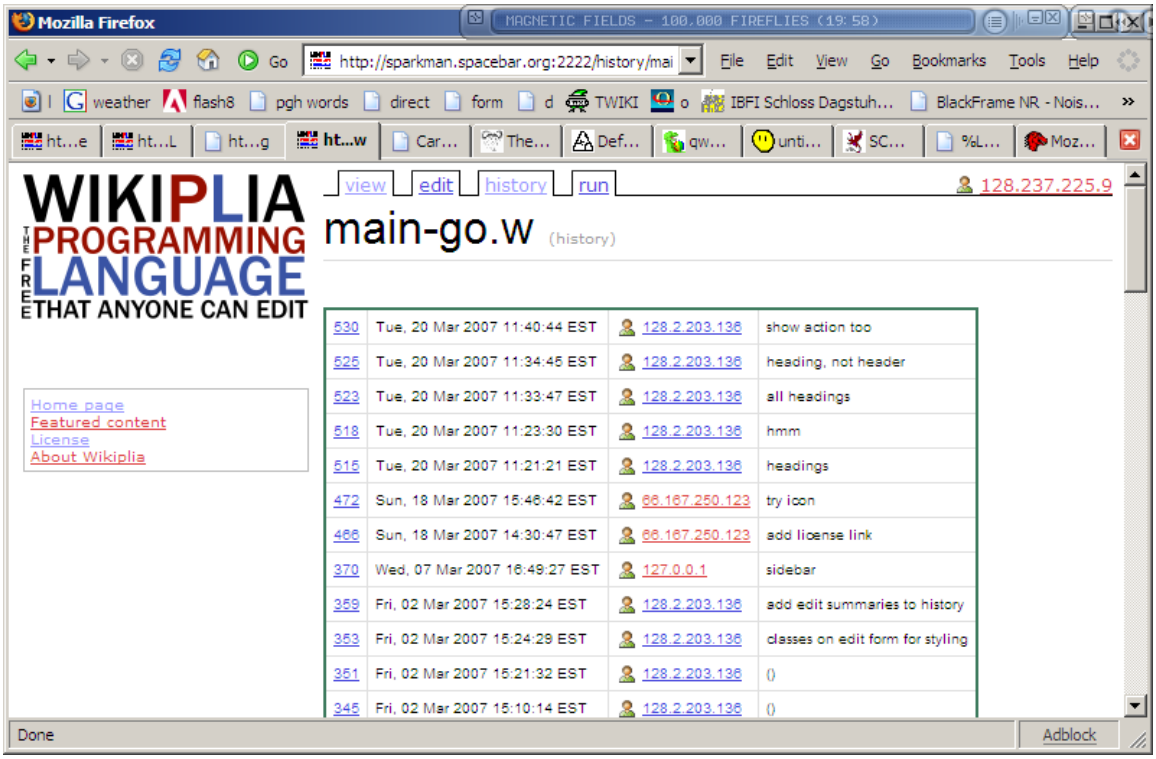


Figure 7: Screenshot of the history information for `main-go.w`.

4.2 Wikiplia language `w`

Writing XESP documents by hand is very tedious, so one of the first orders of business was to develop a compiler for a new language, `w`, which can be extended with convenient features. The first version of this language, created in revision 22, was written in XESP. It automatically quoted the appropriate arguments to the `let`, `it`, `lambda`, `xcase` primitives. In revision 361, the compiler was ported to the language `w` as `w:compile.w`, then compiled with the existing `b`-compiled version of `w:compile.b`, then recompiled with itself until reaching a fixed point.[70] After that, more features were added: a multi-argument function construct `fn`; a list-deconstructing binding construct `lets`; simple support for separately-compiled libraries[71] via `include`; support for mutually-recursive[72] bundles of functions via `fun`;[73] and the `cond` keyword for series of chained "if. . . else" conditionals. At each stage, the feature is implemented using the current version of `w`, and then `w:compile.w` is rewritten to use that convenient extension, and then recompiled until reaching a fixed point.

4.3 Wikiplia language `page`

Wikiplia is not just a programming language; it needs facilities for editing pages that are human readable as well. This can be used to edit documentation for programming languages, to modify the home page to tout new developments, to modify the software license (Section 5) or to deface other user's personal pages. For this we provide a Wiki-like[74] syntax that allows for the authorship of such pages and easy linking between them. Like other Wikis, a link to a page that does not exist is colored red, to alert the user to the opportunity to stake out cyberspace real estate. This syntax is compiled to XESP documents via the `page` language; the resulting documents are active in the sense that they check the status of linked pages to report the correct color on every page load.

[70] Wikipedia, the free encyclopedia: *Fixed point (mathematics)*; 2007
[71] Wikipedia, the free encyclopedia: *Library (computing)*; 2007
[72] Wikipedia, the free encyclopedia: *Recursion (computer science)*; 2007
[73] Prior to this, recursion had to be encoded directly by passing an initial "self" argument to each function.
[74] Wikipedia, the free encyclopedia: *Wiki*; 2007

5 TIA Public License

In this section we reproduce the Total Information Awareness Public License. Commentary is given via footnotes[75] into parts of the license.

TIA PUBLIC LICENSE
Revision 468, March 2007

BEGIN INVINCIBLE SECTION 1

This software is Copyright © 2007–∞
The Regents of the Wikiplia Foundation.
Permission is not granted to reproduce this software or license except by the terms explicitly enumerated below.

I. Invincible Sections

This license contains certain invincible sections, denoted by the text ‘‘BEGIN INVINCIBLE SECTION <n>’’ and ‘‘END INVINCIBLE SECTION <n>’’. Such sections may not be modified under any circumstances.

END INVINCIBLE SECTION 1[76]

BEGIN INVINCIBLE SECTION 2

II. Version Identification and Invalid Licenses

This license must identify itself in the header as Revision <n> for some number <n>, which must be the same as the revision number in the Wikiplia repository for the key ‘‘TPL’’ in which the license text is stored. If this is not the case, then this version of the license is considered Invalid and Void.

Permission is not granted to distribute the software or license using any Invalid version of the license.

END INVINCIBLE SECTION 2[77]

BEGIN INVINCIBLE SECTION 3

III. Option of License

The licensee has the option to choose any revision of the license prior to (numerically less than) this version as the licensing terms for the software and license.

END INVINCIBLE SECTION 3

BEGIN INVINCIBLE SECTION 4

IV. Heredity of License

Any copy or derivative work of this software or license must be licensed under the TIA Public License.

END INVINCIBLE SECTION 4[78]

BEGIN INVINCIBLE SECTION 5

V. Completeness of Copy

This software and license may only be copied in their entirety, including the entire revision history.

END INVINCIBLE SECTION 5[79]

VI. Freedom to edit

Permission is hereby granted to edit this license.[80]

[75] Wikipedia, the free encyclopedia: *footnote*; 2007

[76] Invincible sections exist in order to ensure the sanctity of the license. We first establish that invincible sections will appear and that they are inviolable; this itself is done in an invincible section. Invincible sections cause a limited loss of liberty, but this is the cost of freedom.

[77] This invincible section establishes a connection between license versions and the actual contents of the Wikiplia repository. Note the self-reference: Though this text is in an invincible section and never changes, the referent of “this license” does change as the rest of the license is modified. Because Wikiplia assigns version numbers monotonically, this ensures that the next invincible section is able to guarantee that freedom is monotonic. In the case of an invalid license, no permissions whatsoever are granted, so the licensee *must* use a prior valid version of the license. The initial version of the license is valid.

[78] This clause makes the license “viral” like the GNU GPL, so that freedom is preserved in all descendants of the software.

[79] This section is the centerpiece of Wikiplia; it guarantees that the “source code” to any software or programming language derived from Wikiplia is free from the loopholes described in Section 2 and so maximizes Freedom ©. Note that this does not limit the way that the software can be modified; the programmer might begin by blanking all of the keys he doesn't care about—as long as he preserves the fact that there was *once* something there.

[80] The only non-invincible provision of the original license allows the reader to add provisions that he desires to the license. This guarantees Freedom ¶, the freedom to redefine freedom.

This license is bootstrapping in the sense that it grants only the minimal permissions necessary, after setting up invariants via the behavior-limiting invincible sections. In fact, the original version of the license does not directly permit the licensee to copy the software at all; he must first amend the license using `VI` to give himself this permission.

6 Conclusion

We have reached the end of our journey. But the journey is not complete! We conclude with a discussion of future plans and unrelated work, and then conclude with another paragraph.

6.1 Future Work

Though Wikiplia in its current form is a usable general-purpose programming language, work remains to be done for it to reach its full potential. For one, it needs a vibrant community of contentious and hubristic editors hiding behind anonymous IP addresses boldly asserting half-baked syntactic extensions or enforcing superficial style preferences, gritting their teeth while typing and clicking white knuckled in a kind of road rage[81] created by the dehumanizing semantic markup by which they are forced to communicate.

We also seek to improve the languages. The language `w` needs many more features to speed development: the parenthesis-based XESP syntax should eventually be replaced by a pleasant concrete syntax, if we can get around to it before too much code is written in XESP. A type system[82] is not planned, because type systems restrict Freedom $\hbar$, the freedom of expression. However, we should seek to make Wikiplia as multiparadigm as possible (again, freedom from discrimination on the basis of paradigm[83] orientation), supporting OOPs-oriented programming,[84] aspect-oriented programming,[85] duck-oriented typing,[86] orientation-oriented orienteering,[87] Orient-oriented programming,[88] etc.

The `page` language needs extensions for developing human-readable web pages, mostly for the purpose of creating jazzy graphics and boxes that distract from or directly call attention to obvious problems with the pages without actually addressing those problems.

6.2 Unrelated Work

All popular modern languages are defined via a definitional interpreter[89,90,91] with accompanying O'Reilly "animal" book.[92] The work on Wikiplia is unrelated: We have no animal mascot[93] and the languages are described by a tower of source-to-source translations[94] on top of a universally parseable semantic document in XML[95] form.

The author[96] doesn't think[97] much of musicals,[98] to be perfectly[99] honest,[100] so those are basically[101] a no-go. He also feels that the metric system[102] but paradoxically[103] also time zones[104] are pretty overrated. Ketchup[105] on eggs[106] is gross,[107] but not quite as gross as foie gras[108], which more or less has the word[109] "gross" in its name[110] so duh.[111]

6.3 Another Paragraph

We have described Wikiplia, the free programming language that anyone can edit. Unlike other programming languages, it is designed to support a variety of freedoms (©, $\hbar$, $\triangle$, $\sqrt[x^{1.2}]{\widehat{oo} + \frac{x}{z^2}}$, ¶, ♾) and is explicitly scalable to new freedoms. Wikiplia is implemented in a minimal bootstrapping core based on freedom-aware technologies such as XML, and then built up to a featured system using its own faculties

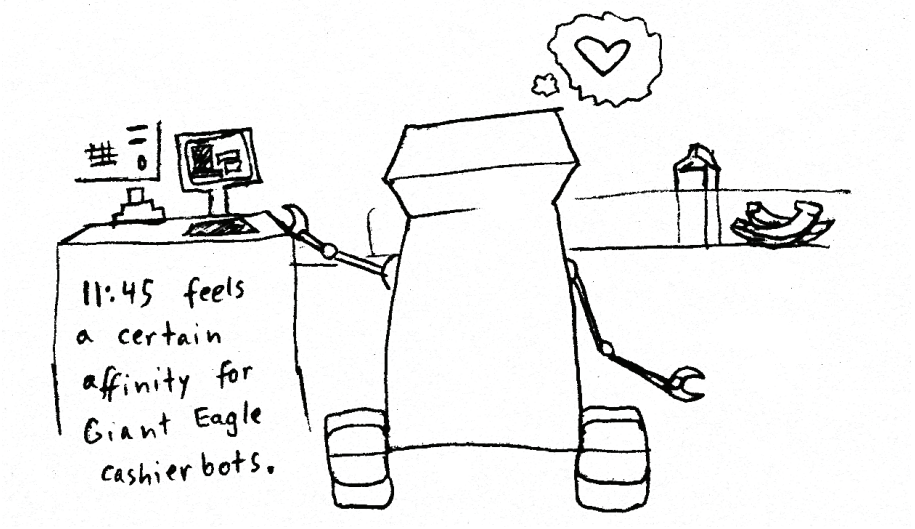
Note that even if a freedom-hater removes this provision from the license, the invincible sections above ensure that the freedom to edit the license is preserved for all time.

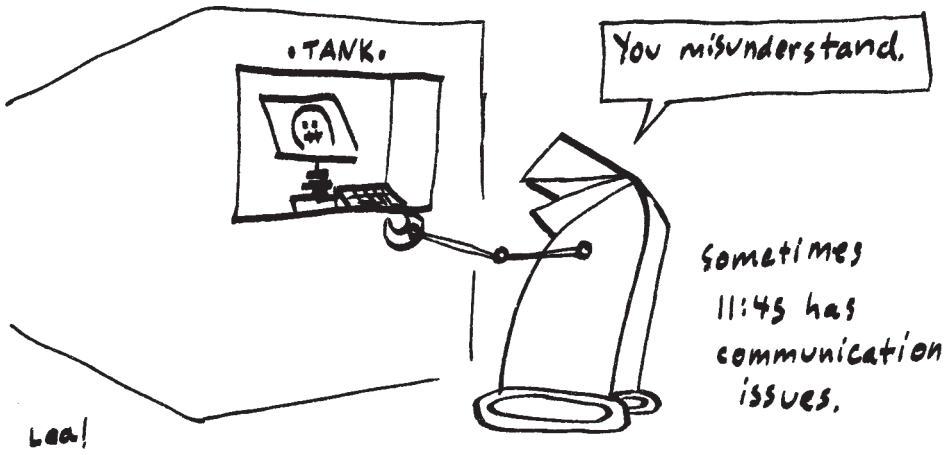
[81] Wikipedia, the free encyclopedia: *List of rages*; 2007
[82] Wikipedia, the free encyclopedia: *Type system*; 2007
[83] Wikipedia, the free encyclopedia: *paradigm*; 2007
[84] Wikipedia, the free encyclopedia: *Object-oriented programming*; 2007
[85] Wikipedia, the free encyclopedia: *Aspect-oriented programming*; 2007
[86] Wikipedia, the free encyclopedia: *Duck typing*; 2007
[87] Wikipedia, the free encyclopedia: *Orienteering*; 2007
[88] Wikipedia, the free encyclopedia: *The Orient*; 2007
[89] Wikipedia, the free encyclopedia: *JavaScript*; 2007
[90] Wikipedia, the free encyclopedia: *Objective Caml*; 2007
[91] Wikipedia, the free encyclopedia: *Perl*; 2007
[92] Wikipedia, the free encyclopedia: *O'Reilly Media*; 2007
[93] Wikipedia, the free encyclopedia: *ORLY owl*; 2007
[94] Wikipedia, the free encyclopedia: *Translation*; 2007
[95] Wikipedia, the free encyclopedia: *Category:ML programming language family*
[96] Wikipedia, the free encyclopedia: *Author*; 2007
[97] Wikipedia, the free encyclopedia: *Thought*; 2007
[98] Wikipedia, the free encyclopedia: *Musical theatre*; 2007
[99] Wikipedia, the free encyclopedia: *Thomas Aquinas*; 2007
[100] Wikipedia, the free encyclopedia: *Honesty*; 2007
[101] Wikipedia, the free encyclopedia: *BASIC*; 2007
[102] Wikipedia, the free encyclopedia: *Metric system*; 2007
[103] Wikipedia, the free encyclopedia: *Pardadox*; 2007
[104] Wikipedia, the free encyclopedia: *Time zone*; 2007
[105] Wikipedia, the free encyclopedia: *Ketchup*; 2007
[106] Wikipedia, the free encyclopedia: *Egg (food)*; 2007
[107] Wikipedia, the free encyclopedia: *Gross*; 2007
[108] Wikipedia, the free encyclopedia: *Foie gras*; 2007
[109] Wikipedia, the free encyclopedia: *Word*; 2007
[110] Wikipedia, the free encyclopedia: *Name*; 2007
[111] Wikipedia, the free encyclopedia: *Duh*; 2007

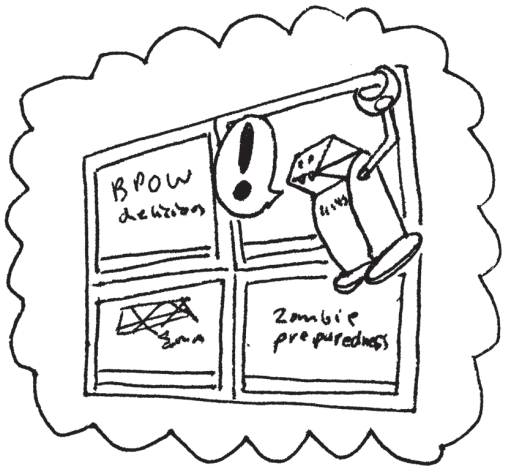
for extension. However, much work remains to be done. We invite you to join us!

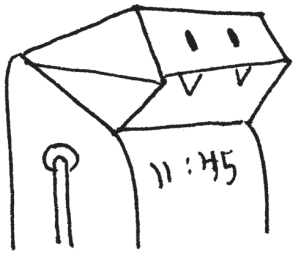
`http://wikiplia.spacebar.org:2222/`

Comics Supplement









11:45 frequently dreams of riding the moving chalkboards.

Lea ©

(N o t e s)