

Verification and Deduction Mentoring Workshop

July 13, 2018

How to Write Research Papers,
or Don't make your
readers *SCREAM!*

Lawrence C Paulson,
Computer Laboratory,
University of Cambridge

Are you guilty of this?

“program testing can show the presence of bugs but never their absence”

Ariane-5 Rocket Explosion



“safety-critical systems blah
blah formal verification”

- ❖ Do you have a three page introduction?
- ❖ Do you give four pages of background theory?
- ❖ Do you include a whole page about future work?
- ❖ Did you copy-paste your abstract, introduction and conclusions?

Why???

Maybe you were imitating
the style of other papers?

Perhaps your research
supervisor has been using the
same clichés for 20 years?

Trying to sound
“academic”?

Scientific writing is *communication!*
— of your *results!*

What people want to know

❖ What did you do?

Introduction

❖ *Why* did you do it?

Background

❖ What **techniques** did you borrow?

Method, etc.

❖ How **precisely** did you do it?

Discussion &
related work

❖ What was the *outcome*?

❖ What did you **discover**?

Conclusions

Title and abstract

- ❖ Your *title* compresses your work down to a few words. It needs to be snappy and clear!

[Readers scan lists of titles in a conference / journal.]

- ❖ Your *abstract* compresses your **entire paper** down to a paragraph.

Main body of paper

- ❖ *Introduction* of the work, including **brief** literature references as motivation
- ❖ *Background*: your starting point; the **minimum** material necessary to read your paper
- ❖ Then **Your methods • Your data • Your analysis**
- ❖ *Discussion* of your results, compared with **related work**
- ❖ *Brief Conclusions* to summarise your **findings**

A Method for Obtaining Digital Signatures and Public-Key Cryptosystems

R. L. Rivest, A. Shamir, and L. Adleman
MIT Laboratory for Computer Science
and Department of Mathematics

(a bit too long!)

Maths should not be typeset as code!

$$E = mc^2 \text{ not } E = m * c^2$$

An encryption method is presented with the novel property that publicly revealing an encryption key does not thereby reveal the corresponding decryption key. This has two important consequences:

- (1) Couriers or other secure means are not needed to transmit keys, since a message can be enciphered using an encryption key publicly revealed by the intended recipient. Only he can decipher the message, since only he knows the corresponding decryption key.
- (2) A message can be “signed” using a privately held decryption key. Anyone can verify this signature using the corresponding publicly revealed encryption key. Signatures cannot be forged, and a signer cannot later deny the validity of his signature. This has obvious applications in “electronic mail” and “electronic funds transfer” systems. A message is encrypted by representing it as a number M , raising M to a publicly specified power e , and then taking the remainder when the result is divided by the publicly specified product, n , of two large secret prime numbers p and q . Decryption is similar; only a different, secret, power d is used, where $e * d \equiv 1 \pmod{(p - 1) * (q - 1)}$. The security of the system rests in part on the difficulty of factoring the published divisor, n .

and the Conclusions

We have proposed a method for implementing a public-key cryptosystem whose security rests in part on the difficulty of factoring large numbers. If the security of our method proves to be adequate, it permits secure communications to be established without the use of couriers to carry keys, and it also permits one to “sign” digitized documents.

The security of this system needs to be examined in more detail. In particular, the difficulty of factoring large numbers should be examined very closely. The reader is urged to find a way to “break” the system. Once the method has withstood all attacks for a sufficient length of time it may be used with a reasonable amount of confidence.

Our encryption function is the only candidate for a “trap-door one-way permutation” known to the authors. It might be desirable to find other examples, to provide alternative implementations should the security of our system turn out someday to be inadequate. There are surely also many new applications to be discovered for these functions.

The abstract and conclusions convey most of the paper’s contributions!

The Next 700 Programming Languages

P. J. Landin

Univac Division of Sperry Rand Corp., New York, New York

A striking title that makes a point

A family of unimplemented computing languages is described that is intended to span differences of application area by a unified framework. This framework dictates the rules about the uses of user-coined names, and the conventions about characterizing functional relationships. Within this framework the design of a specific language splits into two independent parts. One is the choice of written appearances of programs (or more generally, their physical representation). The other is the choice of the abstract entities (such as numbers, character-strings, lists of them, functional relations among them) that can be referred to in the language.

The system is biased towards “expressions” rather than “statements.” It includes a nonprocedural (purely functional) subsystem that aims to expand the class of users’ needs that can be met by a single print-instruction, without sacrificing the important properties that make conventional right-hand-side expressions easy to construct and understand.

Conclusions: Slightly too wordy

11. Conclusion

The languages people use to communicate with computers differ in their intended aptitudes, towards either a particular application area, or a particular phase of computer use (high level programming, program assembly, job scheduling, etc). They also differ in physical appearance, and more important, in logical structure. The question arises, do the idiosyncracies reflect basic logical properties of the situations that are being catered for? Or are they accidents of history and personal background that may be obscuring fruitful developments? This question is clearly important if we are trying to predict or influence language evolution.

To answer it we must think in terms, not of languages, but of families of languages. That is to say we must systematize their design so that a new language is a point chosen from a well-mapped space, rather than a laboriously devised construction.

To this end the above paper has marshalled three techniques of language design: abstract syntax, axiomatization, and an underlying abstract machine.

It is assumed that future calls on language development cannot be forstalled without generalizing the alternatives to explicit sequencing. The innovations of “program-points” and the “off-side rule” are directed at two of the problems (respectively in the areas of semantics and syntax) that must consequently be faced.

Theorems for free!

Philip Wadler

You can get away with a cutesy title if you are a known genius presenting at a high-profile event.
Not otherwise.

From the type of a polymorphic function we can derive a theorem that it satisfies. Every function of the same type satisfies the same theorem. This provides a free source of useful theorems, courtesy of Reynolds' abstraction theorem for the polymorphic lambda calculus.

Admirably concise, but no mention of the key contribution: *parametricity*.

No conclusions, it just ends

The restriction to strict arrows is not to be taken lightly. For instance, given a function r of type

$$r : \forall A. A^* \rightarrow A^*$$

parametricity implies that

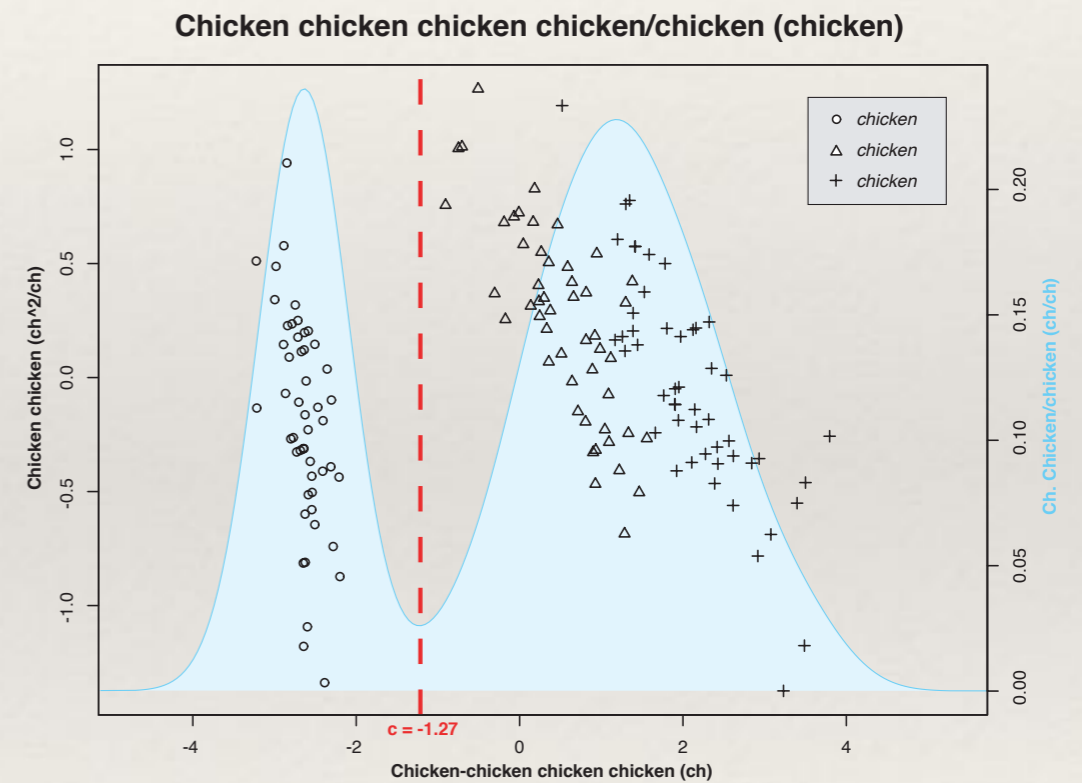
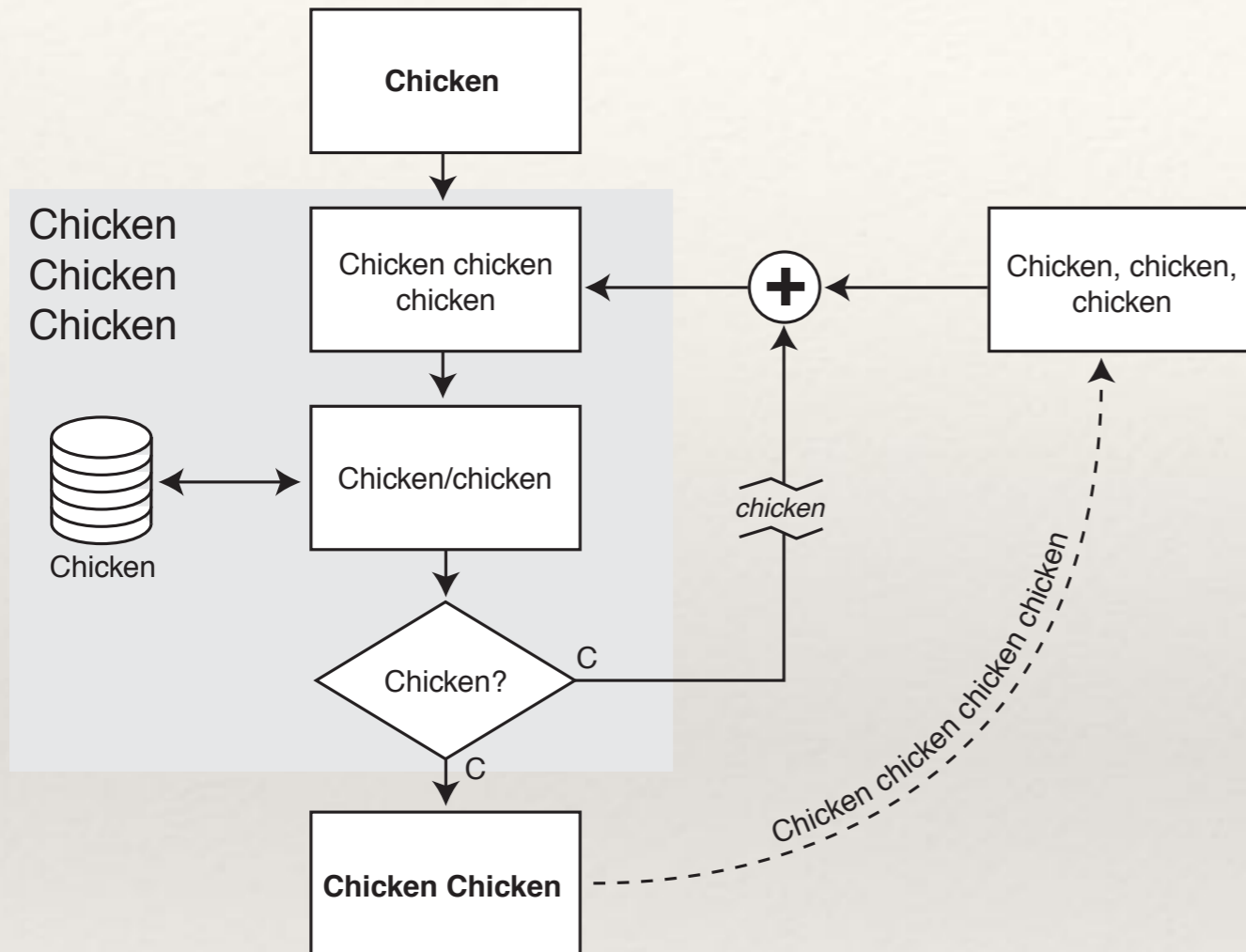
$$r_{A'} \circ a^* = a^* \circ r_A$$

for all functions $a : A \rightarrow A'$. If the fixpoint combinator appears in the definition of r , then we can only conclude that the above holds for strict a , which is a significant restriction.

The desire to derive theorems from types therefore suggests that it would be valuable to explore programming languages that prohibit recursion, or allow only its restricted use. In theory, this is well understood; we have already noted that any computable function that is provably total in second-order Peano arithmetic can be defined in the pure polymorphic lambda calculus, without using the fixpoint as a primitive. However, practical languages based on this notion remain *terra incognita*.

Not a lot of structure in this paper. Still, it has been cited nearly 900 times.

Use diagrams and graphs!



Chicken 1 *Chicken chicken chicken. Chicken chicken, chicken chicken (chicken chicken chicken) chicken chicken-chicken.*

Do tables right!

Not this...

Chicken	CH116	CH116c	CH132
Chicken	57,600	57,600	68,700
Ch. Chicken	2 × 16	2 × 16	2 × 32
Chicken	113:1	113:1	28:1
Ch CCC	0.3463	0.3855	0.3818
chicken	-0.039	0.0050	-0.0015
Ch Ch Chick.	20.91	24.32	24.38
CC	9.08ch	13.30ch	13.40ch

Chicken 1 Chicken chicken chicken. Chicken-chicken chicken chicken chicken chicken chicken chicken, chicken chicken chicken chicken, chicken, chicken chicken chicken “chicken” chicken.

But this!

Table II: Epar, Vampire, and Z3 re-proving with 900 s and Paradox with 30 s (14 185 problems)

ATP	Proved	Unique	Countersat.	Greedy
Vampire	5641	218	0	5641
Epar	5595	194	0	5949
Z3	4375	193	2	6142
Paradox	5	0	2614	6142
Collectively	6142		2614	

See “Publication quality tables in LATEX” by Simon Fear

Briefly: no vertical rules; no double rules; use the **booktabs** package

References

Not this

PA was proved decidable in [17] whose algorithm was improved in [5]. In [12] this procedure was implemented in HOL Light. The general issue of reflection in LCF-like theorem provers is studied in [10].

But this

Presburger [17] proved PA decidable, and Cooper [5] improved his algorithm. Harrison [12] implemented this algorithm in HOL Light; he has also studied [10] the general issue of reflection in LCF-like theorem provers.

Refer to people by *name*, not by *reference number*! Those numbers aren't words.

Be careful with the passive voice! Things are not simply done; people do them.

“I” vs “We”

The problem with “I” is that it makes the paper about *yourself* rather than about *your work*.

Switching to “we” simply turns you into a megalomaniac with delusions of royalty.

“We” can refer to “you and I”, but otherwise keep the focus on objective findings.

Summary

- ❖ The traditional paper structure helps readers *navigate*.
[They may have to read many papers quickly.]
- ❖ Watch for boilerplate such as "outline of the paper", "future work" and long lists of definitions. Don't try to sound "academic"!
- ❖ If you want to be *recognised*, you need to be *understood*!