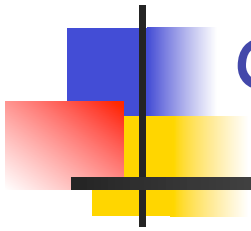


À la recherche des réseaux perdus. 9 décembre 2007. Le tutoriel sera donné par Jon Crowcroft,

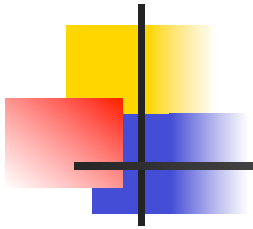
Jon Crowcroft, University of Cambridge

Currently CNRS/LIP6/UPMC

And Thomson Labs, Paris.

Jon.Crowcroft@cl.cam.ac.uk

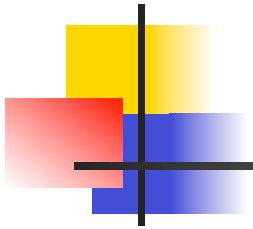
http://www.cl.cam.ac.uk/~jac22



This talk is in English

Unfortunately (desole) my French is insufficient

- 1.4 Billion Indians and Chinese have chosen English (luckily we don't have to learn mandarin or Urdu!)
- Even British have to learn American English for science writing
- We all use Greek for maths anyhow :-)



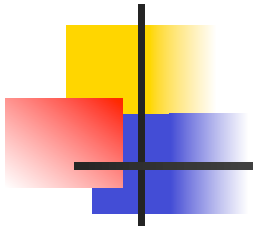
There are 6 parts to this talk

1. Cold Topics in Networks
2. Reading a paper
3. Writing a paper
4. Giving a talk
5. Writing a proposal
6. Hot Topics in Networks

- Acknowledgements to
  - Keshav
  - Simon Peyton Jones
  - Brad Karp and HT Kung
- for materials used with permission.

1. Cold Topics in Networks

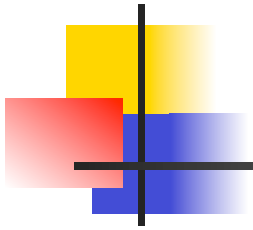
Jon Crowcroft, Cambridge



Hot v. Cold

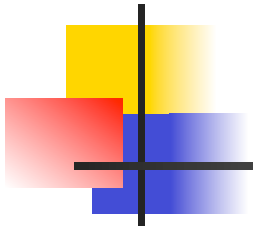
Research goes in cycles - possibly Carnot Cycles

- Topics become hot
- Initially, even, controversial
  - (active nets, social nets, etc)
- Lots of people flock to the topic
- Paradoxically, higher density of researchers reduces temperature
- Topic goes cold.



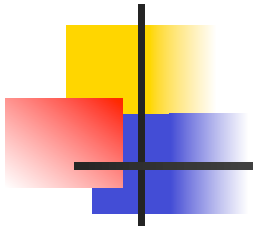
Hot Topics

- Can be detected by brainstorming
  - Socialising and off-the-wall thinking is good
  - Invert a traditional approach
  - Stretch one dimension to an extreme
  - Bisociation/lateral/interdisciplinary
  - Tennenhouse at DARPA (Active Nets) and as head of Intel Research deliberately indulged ***disruptive*** ideas



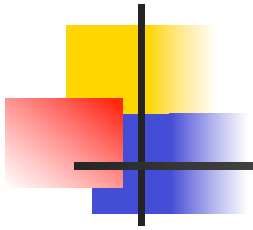
Some measures of cold topics

- Number of low cited papers in low impact conferences
- Fractional Performance delta in systems papers
- Massive uptake of automatic tools for research (NS2, Planetlab, etc)



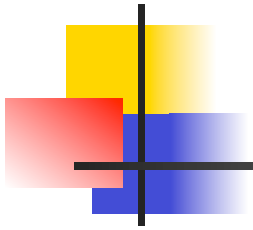
Some examples of cold topics

- [DHT and Structured P2P]
  - Even bad guys like the Storm Botnet use them
- [Internet Coordinate Systems]
  - now secured too!
- [Faster packet classification]
  - If you aren't working with cisco, juniper or huawei?
- [BGP] The Border Gateway Protocol
  - We even have a meta-replacement. Now is the time to deploy.



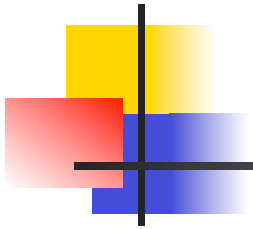
More cold topics

- DDoS
  - Define the problem - DoS on a best effort doesn't mean much - see Newarch.
- Spam
  - Is not largely a technical problem (see social nets and closed user groups) see ddos
- Overlays
  - Were made up as a tool for research, not a research goal!



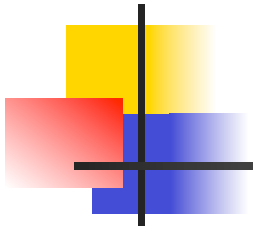
Even more cold topics

- TINA
  - The Intelligent Network Architecture = Knowledge Plane = network management
- TCP+AQM
  - Mostly wrong
- Multicast
  - 20 years without deployment?
- Newarch
  - Not even wrong (see String Theory)



Yet more cold topics

- Self similarity
  - Surely there is a horizon effect
- MANETs
  - 5000 protocols cannto be good
- Self Organising WSNs
  - Unexpected behaviour may not be a plus
- Small World Networks
  - Ask epidemiologists



Now you know what I don't like…

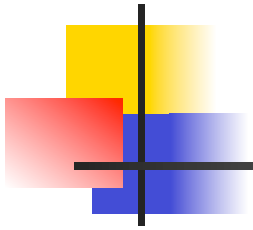
For now :-)

Your PhD topic will have been hot in Year 1. By year 3,4,5 this is unlikely to be still true - consider journals rather than conferences or workshops for later work:-).

2. How to Read a Paper

Jon Crowcroft, Cambridge

Based on CCR Article by Keshav (Waterloo)

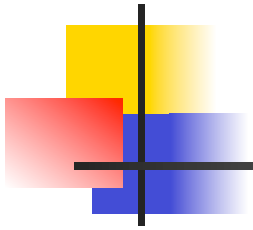


Stand on the Shoulders of Giants

And do not stand on their toes

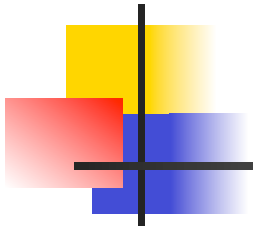
You read other papers so that

- You are learning what papers are like
- You are current in the field
- You may be writing survey (literature review)
- You want to find what to compare with
- We propose a 3 pass reading approach



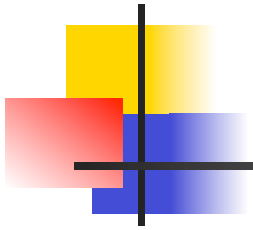
Pass 1

- Structural overview of paper
  - Read abstract/title/intro
  - Read section headings, ignore bodies
  - Read conclusions
  - Scan references noting ones you know



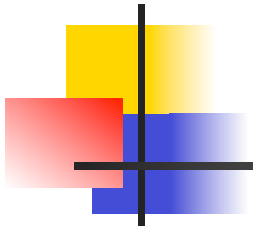
Pass 1 output

- You can now say
  - Is this a system, theory or simulation paper (category defines methodology)
    - Check system measurement methodology
    - Check expressiveness/fit for purpose of formalism
    - Check simulation assumptions
  - What other papers/projects relate to this?
  - Are the assumptions valid?
  - What are the key novel contributions
  - Is the paper clear?
- Takes about 5 minutes
- 95% of reviewers will stop at pass 1 :-(
  - See Section 3 of this (on writing papers)



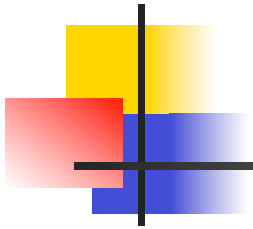
Pass 2

- Check integrity of paper
  - Look at figures/diagrams/exes/definitions
  - Note unfamiliar references
  - Do not check proofs yet
- Takes around 1 hour
- You should be able to summarise the paper to someone else now
- If it is unclear, you may need to pasuse overnight



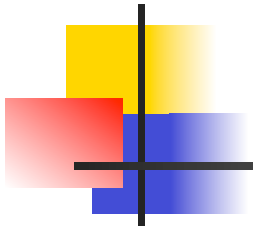
Pass 3

- Virtually re-implement the paper
  - Challenge all assumptions
  - Think adversarially about experiments, proofs, simulation scenarios
  - Takes 4-5 hours
- You should be able to reconstruct paper completely now



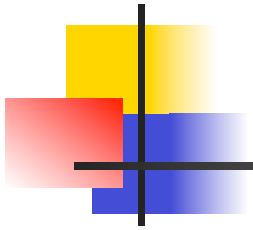
Reading batches of papers

- E.g. for literature survey excercise
  - pick topic (hot or cold), and search on google scholar or citeseer for 10 top papers
  - Find shared citations and repeated author names - key papers (look at citation count/ impact too)
  - Go to venues for these papers and look at other papers



See also

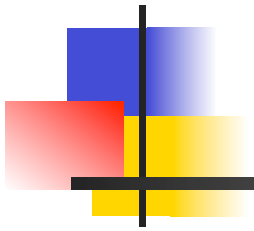
- Timothy Roscoe's
  - Writing reviews for Systems Conferences
- Writing Technical Articles
  - Henning Schulzrinne's



Now you can review papers…

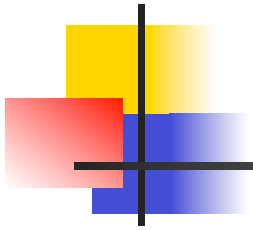
For me:)

You will read 100 papers to every one you write. 90 of them will be much worse, some will be better. A few the same.



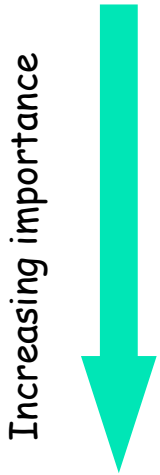
3. How to write a great research paper

Simon Peyton Jones
Microsoft Research, Cambridge



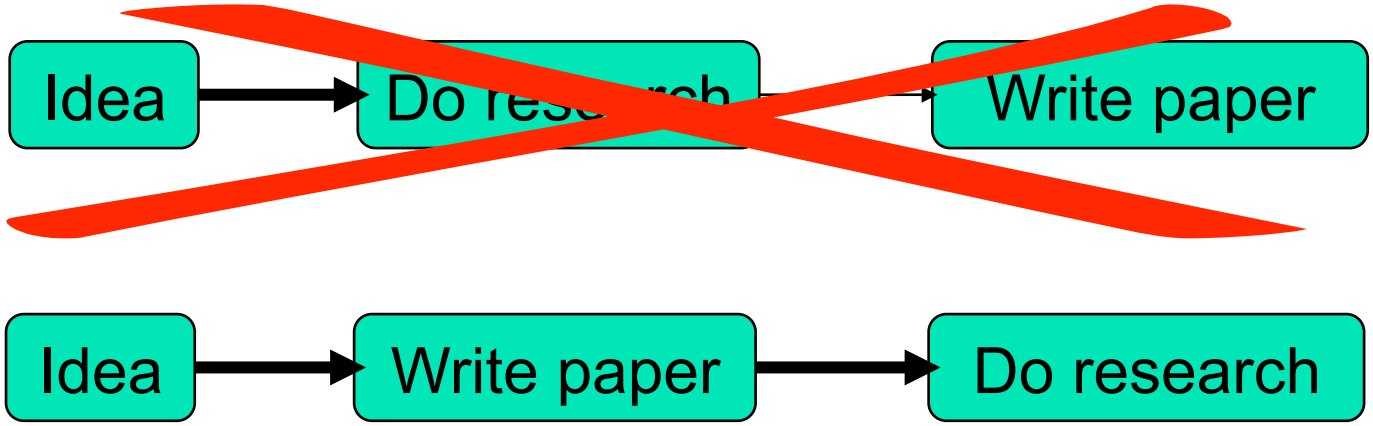
Writing papers is a skill

- Many papers are badly written
- Good writing is a skill you can learn
- It's a skill that is worth learning:
  - You will get more brownie points (more papers accepted etc)
  - Your ideas will have more impact
  - You will have better ideas

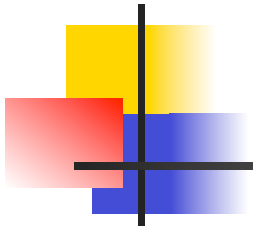


Writing papers: model 1

Writing papers: model 2



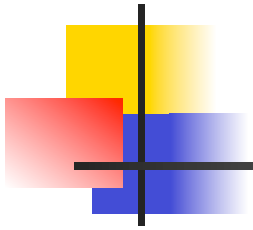
- Forces us to be clear, focused
- Crystallises what we don't understand
- Opens the way to dialogue with others: reality check, critique, and collaboration



Do not be intimidated

Fallacy You need to have a fantastic idea before you can write a paper. (Everyone else seems to.)

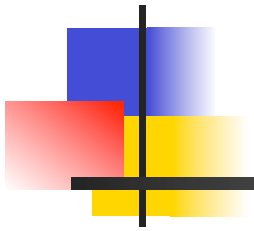
Write a paper,
and give a talk, about
any idea,
no matter how weedy and insignificant it may seem to you



Do not be intimidated

Write a paper, and give a talk, about any idea, no matter how insignificant it may seem to you

- **Writing the paper is how you develop the idea in the first place**
- It usually turns out to be more interesting and challenging that it seemed at first

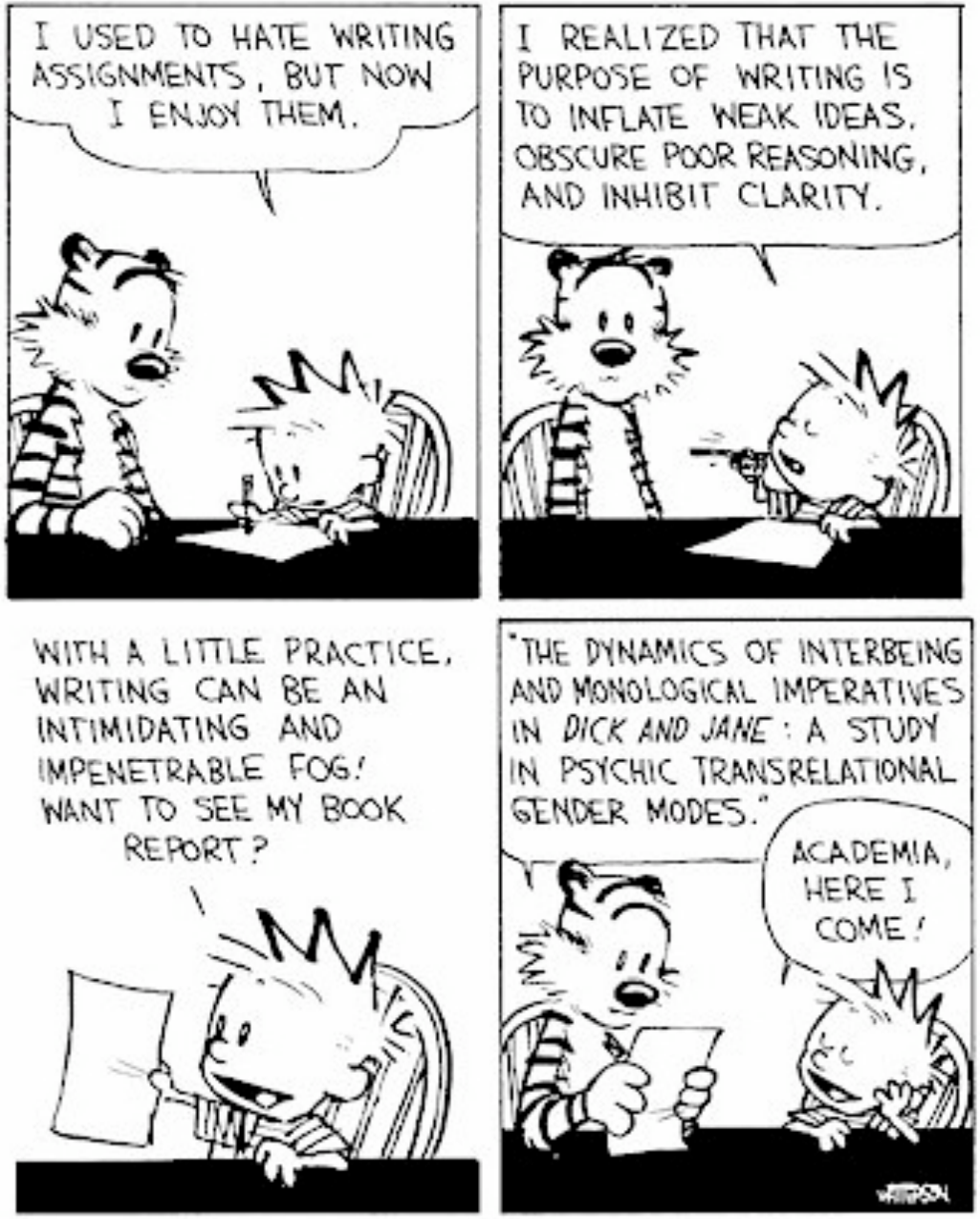


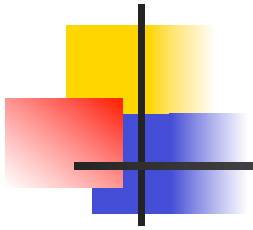
The purpose of your paper

Why bother?

Fallacy

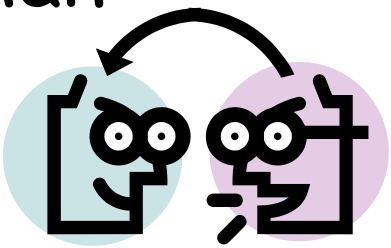
we write papers and give talks mainly to impress others, gain recognition, and get promoted



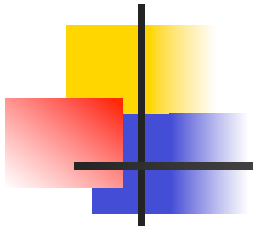


Papers communicate ideas

- Your goal: to infect the mind of your reader with **your idea**, like a virus
- Papers are far more durable than programs (think Mozart)



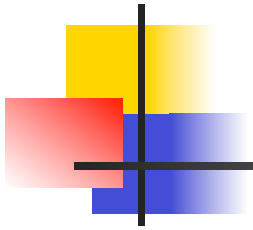
The greatest ideas are (literally) worthless if you keep them to yourself



The Idea

Idea
A re-usable insight, useful to the reader

- Figure out what your idea is
- Make certain that the reader is in no doubt what the idea is. Be 100% explicit:
  - "The main idea of this paper is...."
  - "In this section we present the main contributions of the paper."
- Many papers contain good ideas, but do not distil what they are.



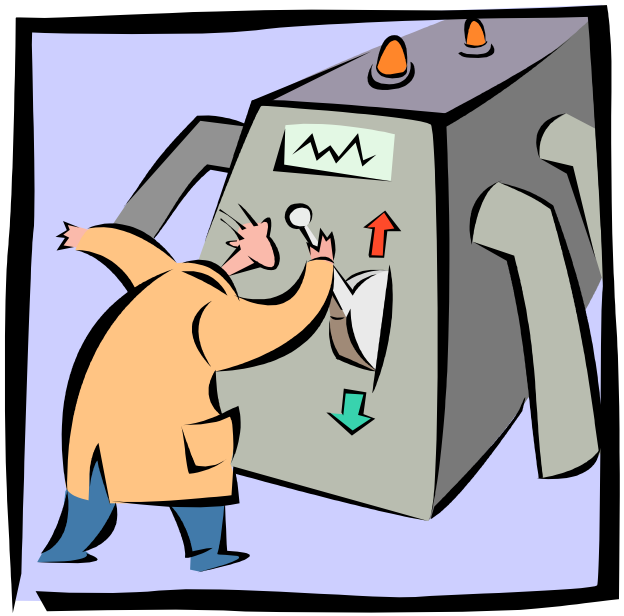
One ping

- Your paper should have just one "ping": one clear, sharp idea
- Read your paper again: can you hear the "ping"?
- You may not know exactly what the ping is when you start writing; but you must know when you finish
- If you have lots of ideas, write lots of papers

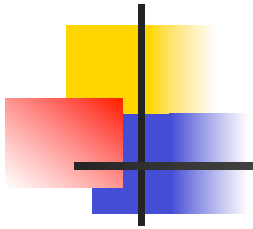
Thanks to Joe Touch for "one ping"

The purpose of your paper is not...

To describe the WizWoz system



- Your reader does not have a WizWoz
- She is primarily interested in re-usable brain-stuff, not executable artefacts

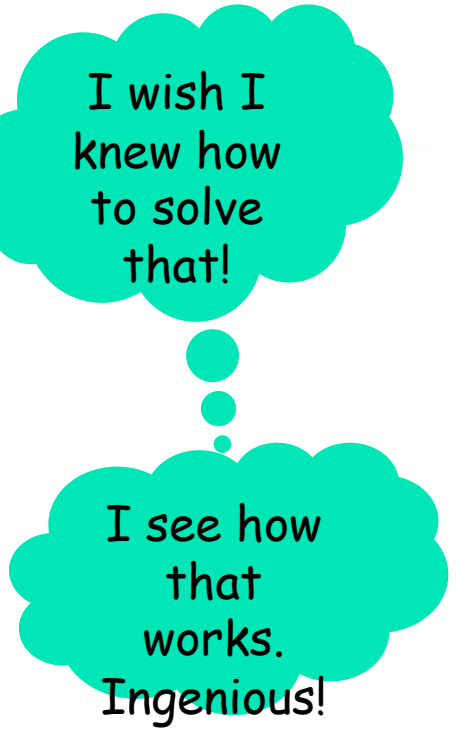


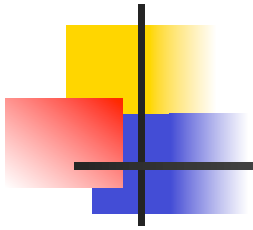
Examples of WizWoz

- Crash Proof OS for Mobile Phones (singularity in F# on an iPhone)
- Go Faster VM (Xen)
- Nimrod

Your narrative flow

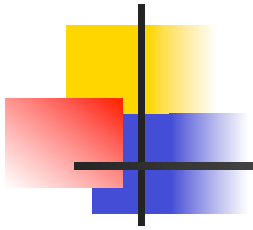
- Here is a problem
- It's an interesting problem
- It's an unsolved problem
- **Here is my idea**
- My idea works (details, data)
- Here's how my idea compares to other people's approaches





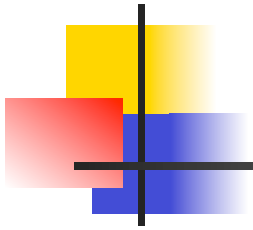
Structure (conference paper)

- Title (1000 readers)
- Abstract (4 sentences, 100 readers)
- Introduction (1 page, 100 readers)
- The problem (1 page, 10 readers)
- My idea (2 pages, 10 readers)
- The details (5 pages, 3 readers)
- Related work (1-2 pages, 10 readers)
- Conclusions and further work (0.5 pages)
  - See section 2 (on reading!)



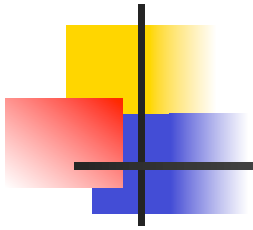
The abstract

- I usually write the abstract last
- Used by program committee members to decide which papers to read
- Four sentences [Kent Beck]
  1. State the problem
  2. Say why it's an interesting problem
  3. Say what your solution achieves
  4. Say what follows from your solution



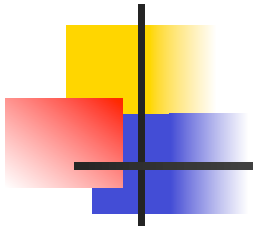
Example

1. Many papers are badly written and hard to understand
2. This is a pity, because their good ideas may go unappreciated
3. Following simple guidelines can dramatically improve the quality of your papers
4. Your work will be used more, and the feedback you get from others will in turn improve your research



Structure

- Abstract (4 sentences)
- **Introduction** (1 page)
- The problem (1 page)
- My idea (2 pages)
- The details (5 pages)
- Related work (1-2 pages)
- Conclusions and further work (0.5 pages)

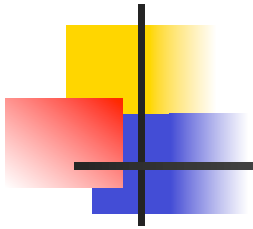


The introduction (1 page)

1. **Describe the problem**
2. **State your contributions**

...and that is all

ONE PAGE!



Describe the problem

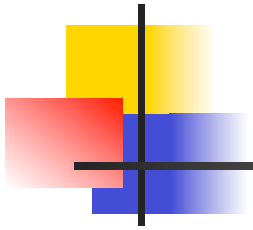
1 Introduction

There are two basic ways to implement function application in a higher-order language, when the function is unknown: the *push/enter* model or the *eval/apply* model [11]. To illustrate the difference, consider the higher-order function **zipWith**, which zips together two lists, using a function **k** to combine corresponding list elements:

```
zipWith :: (a->b->c) -> [a] -> [b] -> [c]
zipWith k []     []     = []
zipWith k (x:xs) (y:ys) = k x y : zipWith xs ys
```

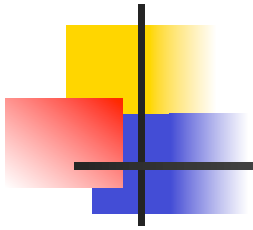
Here **k** is an *unknown function*, passed as an argument; global flow analysis aside, the compiler does not know what function **k** is bound to. How should the compiler deal with the call **k x y** in the body of **zipWith**? It can't blithely apply **k** to two arguments, because **k** might in reality take just one argument and compute for a while before returning a function that consumes the next argument; or **k** might take three arguments, so that the result of the **zipWith** is a list of functions.

Use an example to introduce the problem



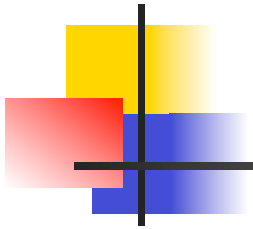
e.g. of systems problem

- Mobile Phones crash a lot
- Wireless media is vulnerable
- Bad software on mobile phone can hurt user (cost money, time, pain)
- Bad software on radio can hurt all users
- We have a lot better tools to write safer software and have done so on desktops and servers
- Can they work on small devices with limited resources, and if so, how well?



State your contributions

- Write the list of contributions first
- The list of contributions drives the entire paper: the paper substantiates the claims you have made
- Reader thinks "gosh, if they can really deliver this, that's be exciting; I'd better read on"



State your contributions

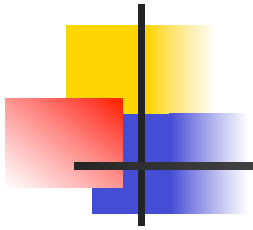
Which of the two is best in practice? The trouble is that the evaluation model has a pervasive effect on the implementation, so it is too much work to implement both and pick the best. Historically, compilers for strict languages (using call-by-value) have tended to use eval/apply, while those for lazy languages (using call-by-need) have often used push/enter, but this is 90% historical accident — either approach will work in both settings. In practice, implementors choose one of the two approaches based on a qualitative assessment of the trade-offs. In this paper we put the choice on a firmer basis:

- We explain precisely what the two models are, in a common notational framework (Section 4). Surprisingly, this has not been done before.
- The choice of evaluation model affects many other design choices in subtle but pervasive ways. We identify and discuss these effects in Sections 5 and 6, and contrast them in Section 7. There are lots of nitty-gritty details here, for which we make no apology — they were far from obvious to us, and articulating these details is one of our main contributions.

  In terms of its impact on compiler and run-time system complexity, eval/apply seems decisively superior, principally because push/enter requires a stack like no other: stack-walking

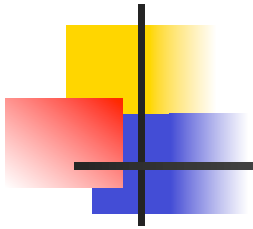
Bulleted list of contributions

Do not leave the reader to guess what your contributions are!



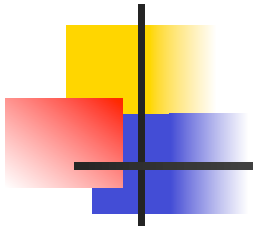
E.g. of systems contributions

- We encapsulate all the modules of software on a cell phone in F# behavioural description wrappers, and
- Run a model checker on them (e.g. isobel)
- And then try various well known attacks that fail on desk top but succeed on windows mobile and symbian phones
- We then show our software is also smaller and faster….



Contributions should be refutable

NO!	YES!
We describe the WizWoz system. It is really cool.	We give the syntax and semantics of a language that supports concurrent processes (Section 3). Its innovative features are...
We study its properties	We prove that the type system is sound, and that type checking is decidable (Section 4)
We have used WizWoz in practice	We have built a GUI toolkit in WizWoz, and used it to implement a text editor (Section 5). The result is half the length of the Java version.

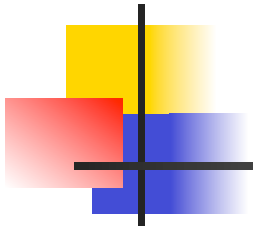


No "rest of this paper is..."

- Not:

> "The rest of this paper is structured as follows. Section 2 introduces the problem. Section 3 ... Finally, Section 8 concludes".

- Instead, **use forward references from the narrative in the introduction**. The introduction (including the contributions) should survey the whole paper, and therefore forward reference every important part.

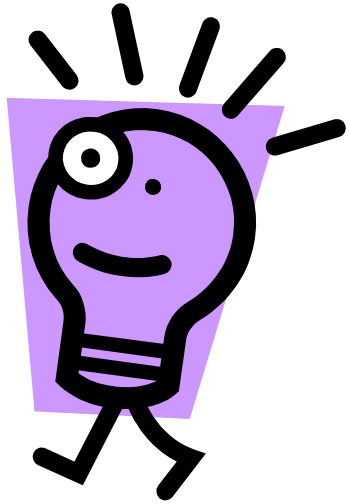


Structure

- Abstract (4 sentences)
- Introduction (1 page)
- ~~Related work~~
- The problem (1 page)
- My idea (2 pages)
- The details (5 pages)
- Related work (1-2 pages)
- Conclusions and further work (0.5 pages)

No related work yet!

Related work



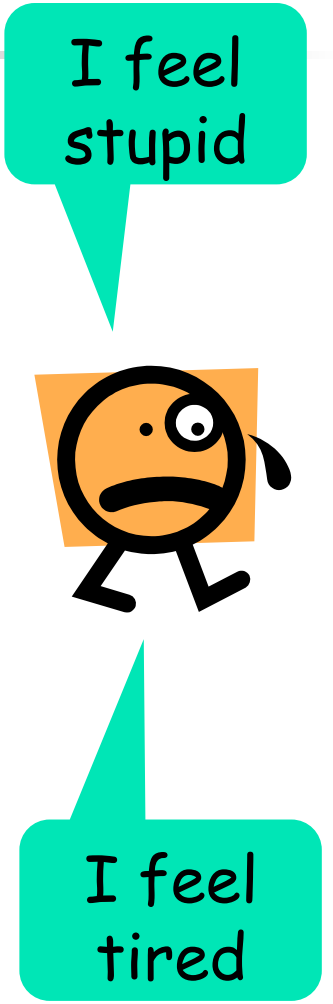
Your reader

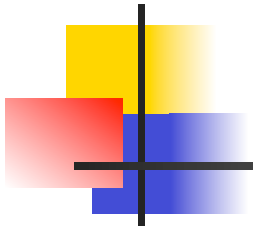
Your idea

We adopt the notion of transaction from Brown [1], as modified for distributed systems by White [2], using the four-phase interpolation algorithm of Green [3]. Our work differs from White in our advanced revocation protocol, which deals with the case of priority inversion as described by Yellow [4].

No related work yet

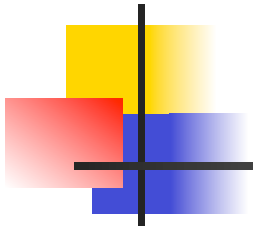
- Problem 1: the reader knows nothing about the problem yet; so your (carefully trimmed) description of various technical tradeoffs is absolutely incomprehensible
- Problem 2: describing alternative approaches gets between the reader and your idea





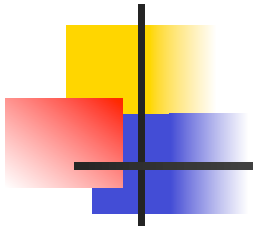
Related work and survey

- Obviously, if your paper is a survey…
- …then it is all related work☺
- Gap analysis is sometimes useful in an introduction, but it is not quite the same as related work
- Taxonomies…are quite handy in that case



Structure

- Abstract (4 sentences)
- Introduction (1 page)
- The problem (1 page)
- My idea (2 pages)
- The details (5 pages)
- Related work (1-2 pages)
- Conclusions and further work (0.5 pages)

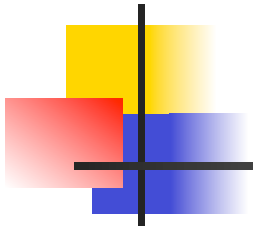


Presenting the idea

3. The idea

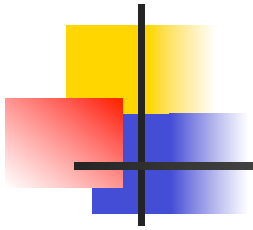
Consider a bifircuated semi-lattice D, over a hyper-modulated signature S. Suppose p_i is an element of D. Then we know for every such p_i there is an epi-modulus j, such that $p_j < p_i$.

- Sounds impressive...but
- Sends readers to sleep
- In a paper you MUST provide the details, but FIRST convey the idea



Presenting the idea

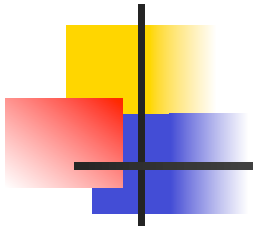
- Explain it as if you were speaking to someone using a whiteboard
- **Conveying the intuition is primary**, not secondary
- Once your reader has the intuition, she can follow the details (but not vice versa)
- Even if she skips the details, she still takes away something valuable



Putting the reader first

- **Do not** recapitulate your personal journey of discovery. This route may be soaked with your blood, but that is not interesting to the reader.

- Instead, choose the most direct route to the idea.



The payload of your paper

Introduce the problem, and your idea, using

EXAMPLES

and only then present the general case

Using examples

The Simon PJ question: is there any typewriter font?

2 Background

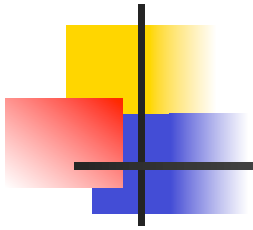
To set the scene for this paper, we begin with a brief overview of the *Scrap your boilerplate* approach to generic programming. Suppose that we want to write a function that computes the size of an arbitrary data structure. The basic algorithm is "for each node, add the sizes of the children, and add 1 for the node itself". Here is the entire code for `gsize`:

```
gsize :: Data a => a -> Int
gsize t = 1 + sum (gmapQ gsize t)
```

The type for `gsize` says that it works over any type `a`, provided `a` is a *data* type — that is, that it is an instance of the class `Data`[1] The definition of `gsize` refers to the operation `gmapQ`, which is a method of the `Data` class:

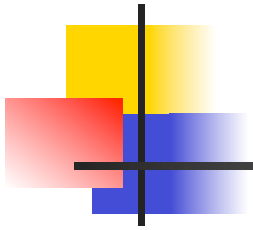
```
class Typeable a => Data a where
  ...other methods of class Data...
  gmapQ :: (forall b. Data b => b -> r) -> a -> [r]
```

Example right away



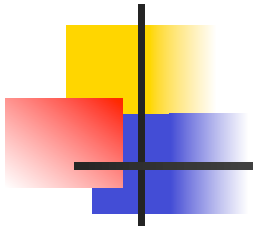
The details: evidence

- Your introduction makes claims
- The body of the paper provides **evidence to support each claim**
- Check each claim in the introduction, identify the evidence, and forward-reference it from the claim
- Evidence can be: analysis and comparison, theorems, measurements, case studies



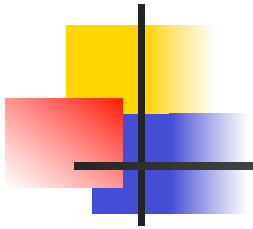
In my safeOS for handset e.g.

- One would give code fragments of unsafe code
- And examples of threats…
- …and examples of safe code
- And some performance results…
- Before launching in to the description of the small fast efficient compile and runtime checks possible in new system…



Structure

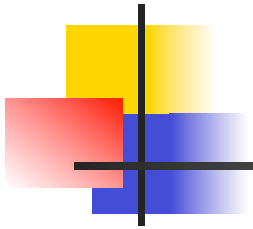
- Abstract (4 sentences)
- Introduction (1 page)
- The problem (1 page)
- My idea (2 pages)
- The details (5 pages)
- **Related work** (1-2 pages)
- Conclusions and further work (0.5 pages)



Related work

Fallacy

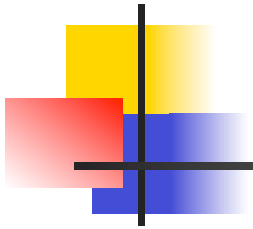
To make my work look good, I have to make other people's work look bad



The truth: credit is not like money

Giving credit to others does not diminish the credit you get from your paper

- Warmly acknowledge people who have helped you
- Be generous to the competition. "In his inspiring paper [Foo98] Foogle shows.... We develop his foundation in the following ways..."
- Acknowledge weaknesses in your approach

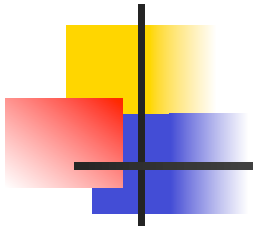


Credit is not like money

Failing to give credit to others can kill your paper

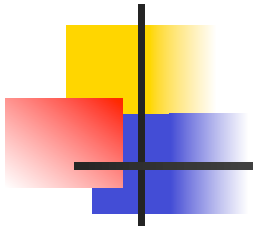
If you imply that an idea is yours, and the referee knows it is not, then either

- You don't know that it's an old idea (bad)
- You do know, but are pretending it's yours (very bad)



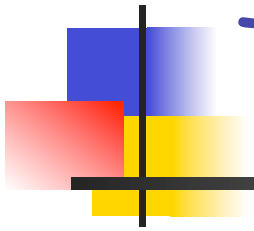
Structure

- Abstract (4 sentences)
- Introduction (1 page)
- The problem (1 page)
- My idea (2 pages)
- The details (5 pages)
- Related work (1-2 pages)
- Conclusions and further work (0.5 pages)

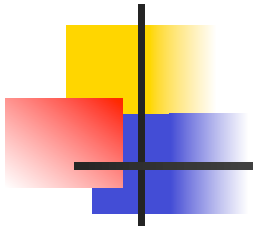


Conclusions and further work

- Be brief.

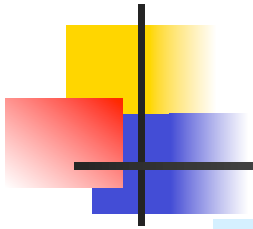


The process of writing



The process

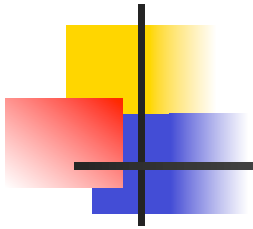
- Start early. Very early.
  - Hastily-written papers get rejected.
  - Papers are like wine: they need time to mature
- Collaborate
- Use CVS to support collaboration



Getting help

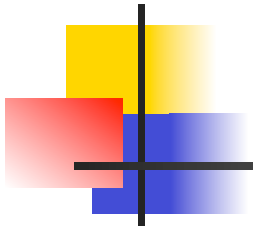
Get your paper read by as many friendly guinea pigs as possible

- Experts are good
- Non-experts are also very good
- Each reader can only read your paper for the first time once! So use them carefully
- Explain carefully what you want ("I got lost here" is much more important than "Jarva is mis-spelt".)



Getting expert help

- A good plan: when you think you are done, send the draft to the competition saying "could you help me ensure that I describe your work fairly?".
- Often they will respond with helpful critique (they are interested in the area)
- They are likely to be your referees anyway, so getting their comments or criticism up front is Jolly Good.



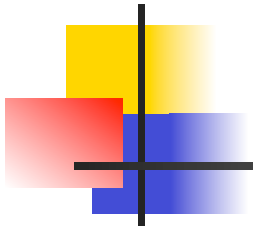
Listening to your reviewers

Treat every review like gold dust

Be (truly) grateful for criticism as well as praise

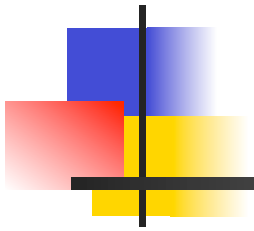
This is really, really, really hard

But it's
really, really, really, really, really, really, really, really, really, really
important

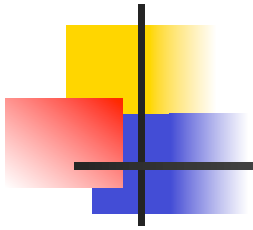


Listening to your reviewers

- Read every criticism as a positive suggestion for something you could explain more clearly
- DO NOT respond "you stupid person, I meant X". Fix the paper so that X is apparent even to the stupidest reader.
- Thank them warmly. They have given up their time for you.

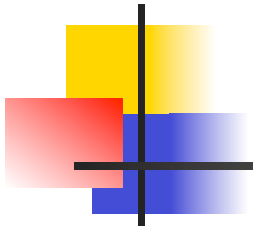


Language and style



Basic stuff

- Submit by the deadline
- Keep to the length restrictions
  - Do not narrow the margins
  - Do not use 6pt font
  - On occasion, supply supporting evidence (e.g. experimental data, or a written-out proof) in an appendix
- Always use a spell checker



Visual structure

- Give strong visual structure to your paper using
  - sections and sub-sections
  - bullets
  - italics
  - laid-out code
- Find out how to draw pictures, and use them

Visual structure

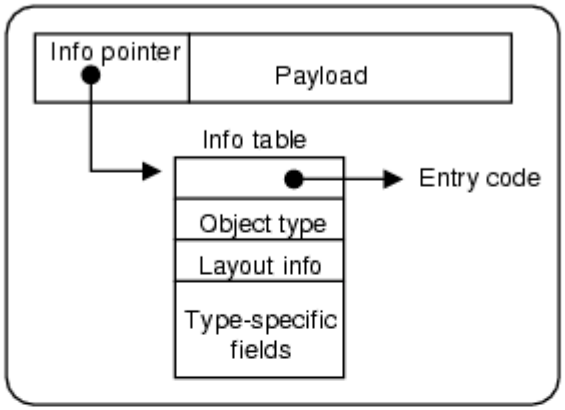


Figure 3. A heap object

The three cases above do not exhaust the possible forms of f. It might also be a *THUNK*, but we have already dealt with that case (rule THUNK). It might be a *CON*, in which case there cannot be any pending arguments on the stack, and rules UPDATE or RET apply.

4.3 The eval/apply model

The last block of Figure 2 shows how the eval/apply model deals with function application. The first three rules all deal with the case of a *FUN* applied to some arguments:

- If there are exactly the right number of arguments, we behave exactly like rule KNOWNCALL, by tail-calling the function. Rule EXACT is still necessary — and indeed has a direct counterpart in the implementation — because the function might not be statically known.
- If there are too many arguments, rule CALLK pushes a *call*

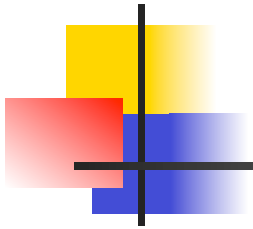
remainder of the object is called the *payload*, and may consist of a mixture of pointers and non-pointers. For example, the object $CON(C\ a_1 \ldots a_n)$ would be represented by an object whose info pointer represented the constructor C and whose payload is the arguments $a_1 \ldots a_n$.

The info table contains:

- Executable code for the object. For example, a *FUN* object has code for the function body.
- An object-type field, which distinguishes the various kinds of objects (*FUN*, *PAP*, *CON* etc) from each other.
- Layout information for garbage collection purposes, which describes the size and layout of the payload. By "layout" we mean which fields contain pointers and which contain non-pointers, information that is essential for accurate garbage collection.
- Type-specific information, which varies depending on the object type. For example, a *FUN* object contains its arity; a *CON* object contains its constructor tag, a small integer that distinguishes the different constructors of a data type; and so on.

In the case of a PAP, the size of the object is not fixed by its info table; instead, its size is stored in the object itself. The layout of its fields (e.g. which are pointers) is described by the (initial segment of) an argument-descriptor field in the info table of the FUN object which is always the first field of a PAP. The other kinds of heap object all have a size that is statically fixed by their info table.

A very common operation is to jump to the entry code for the object, so GHC uses a slightly-optimised version of the representation in Figure 3. GHC places the info table at the addresses *immediately*



Use the active voice

The passive voice is "respectable" but it DEADENS your paper. Avoid it at all costs.

NO	YES
It can be seen that...	We can see that...
34 tests were run	We ran 34 tests
These properties were thought desirable	We wanted to retain these properties
It might be thought that this would be a type error	You might think this would be a type error

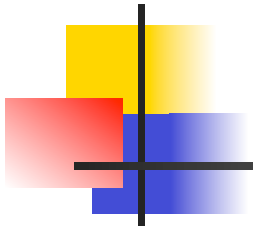
"We" = you and the reader

"We" = the authors

"You" = the reader

Use simple, direct language

NO	YES
The object under study was displaced horizontally	The ball moved sideways
On an annual basis	Yearly
Endeavour to ascertain	Find out
It could be considered that the speed of storage reclamation left something to be desired	The garbage collector was really slow



Summary

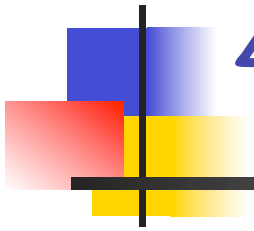
If you remember nothing else:

- **Identify your key idea**
- **Make your contributions explicit**
- **Use examples**

A good starting point:

"Advice on Research and Writing"

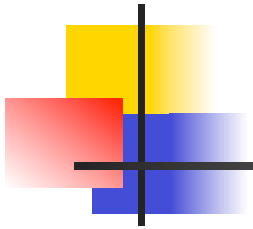
http://www-2.cs.cmu.edu/afs/cs.cmu.edu/user/mleone/web/how-to.html



4. How to give a good research talk

Simon Peyton Jones
Microsoft Research, Cambridge

1993 paper joint with
John Hughes (Chalmers),
John Launchbury (Oregon Graduate Institute)

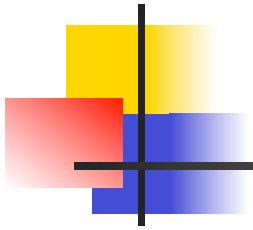


Research is communication

The greatest ideas are worthless if you keep them to yourself

Your papers and talks

- Crystalise your ideas
- Communicate them to others
- Get feedback
- Build relationships
- (And garner research brownie points)



Do it! Do it! Do it!

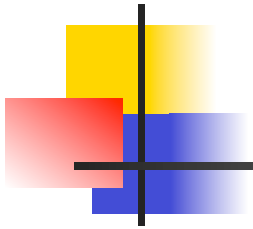
Good papers and talks are a fundamental part of research excellence

- Invest time
- Learn skills
- Practice

Write a paper, and give a talk, about

any idea,

no matter how weedy and insignificant it may seem to you



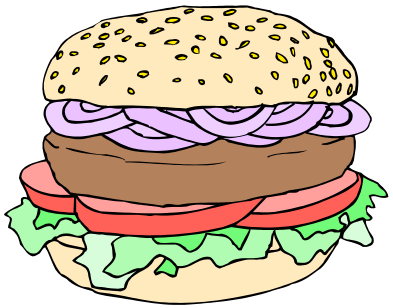
Giving a good talk

This presentation is about how to give a good research talk

- What your talk is for
- What to put in it (and what not to)
- How to present it

What your talk is for

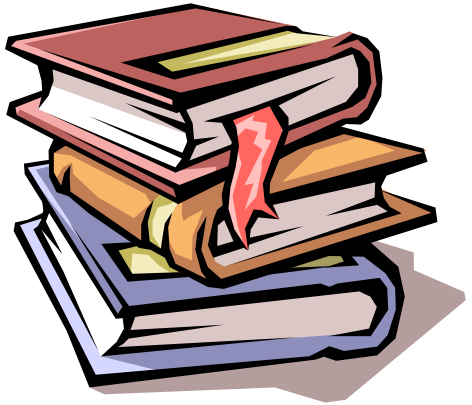
Your paper = **The beef**



Your talk = **The beef advertisment**

Do not confuse the two, even if you are vegetarian

The purpose of your talk…



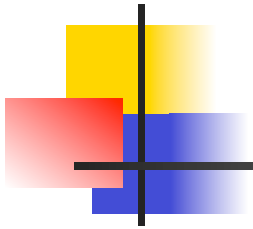
..is not:

- To impress your audience with your brainpower
- To tell them all you know about your topic
- To present all the technical details

The purpose of your talk…

..but is:

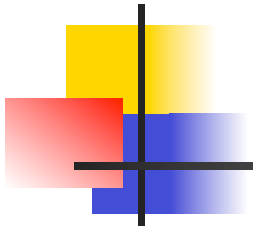
- To give your audience an intuitive feel for your idea
- To make them foam at the mouth with eagerness to read your paper
- To engage, excite, provoke them



Your audience…

The audience you would like

- Have read all your earlier papers
- Thoroughly understand all the relevant theory of cartesian closed endomorphic bifunctors
- Are all agog to hear about the latest developments in your work
- Are fresh, alert, and ready for action



Your actual audience…

The audience you get

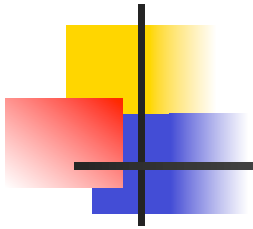
- Have never heard of you
- Have heard of bifunctors, but wish they hadn't
- Have just had lunch or been skiing for 5 hours and are ready for a doze

Your mission is to

WAKE THEM UP

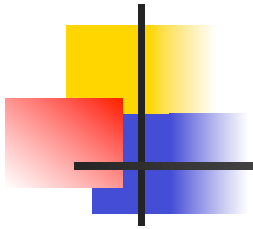
And make them glad they did

What to put in



What to put in

1. Motivation (20%)
2. Your key idea (80%)
3. There is no 3



Motivation

You have 2 minutes to engage your audience before they start to doze

- Why should I tune into this talk?
- What is the problem?
- Why is it an interesting problem?

Example: Java class files are large (brief figures), and get sent over the network. Can we use language-aware compression to shrink them?

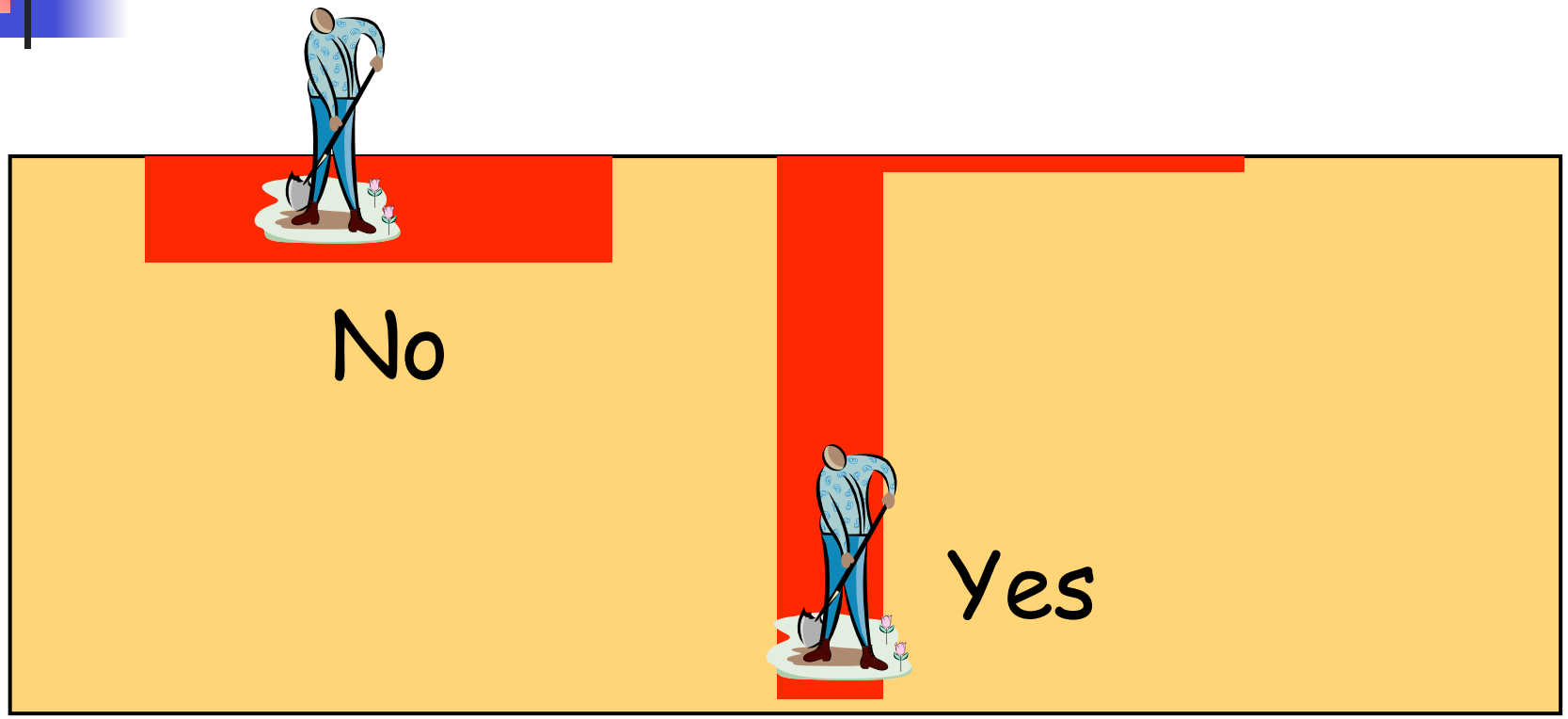
Example: synchronisation errors in concurrent programs are a nightmare to find. I'm going to show you a type system that finds many such errors at compile time.

Your key idea

If the audience remembers only one thing from your talk, what should it be?

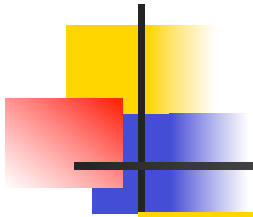
- You must identify a key idea. "What I did this summer" is No Good.
- Be specific. Don't leave your audience to figure it out for themselves.
- Be absolutely specific. Say "If you remember nothing else, remember this."
- Organise your talk around this specific goal. Ruthlessly prune material that is irrelevant to this goal.

Narrow, deep beats wide, shallow



Avoid shallow overviews at all costs

Cut to the chase: the technical "meat"

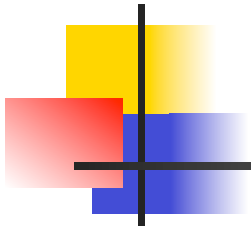


Your main weapon

Examples are your main weapon

- To motivate the work
- To convey the basic intuition
- To illustrate The Idea in action
- To show extreme cases
- To highlight shortcomings

When time is short, omit the general case, not the example



Exceptions in Haskell?

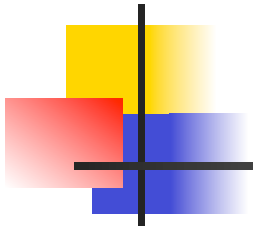
Exceptions are to do with control flow
There is no control flow in a lazy functional program

Solution 1: use data values to carry exceptions

```
data Maybe a = Nothing
             | Just a

lookup :: Name -> Dictionary -> Maybe Address
```

Often this is Just The Right Thing
[Spivey 1990, Wadler "list of successes"]



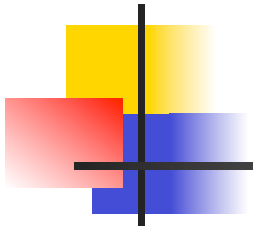
Another example

- Virtual diseases on handsets
- Give SIR equation
- Look at real population
- Compare range of S, I and R parameters
- and complexity of mix networks..

What to leave out

Outline of my talk

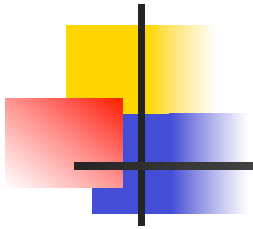
- Background
- The FLUGOL system
- Shortcomings of FLUGOL
- Overview of synthetic epimorphisms
- π-reducible decidability of the pseudo-curried fragment under the Snezkovwski invariant in FLUGOL
- Benchmark results
- Related work
- Conclusions and further work



No outline!

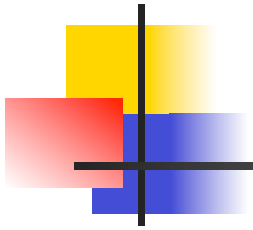
"Outline of my talk": conveys near zero information at the start of your talk

- But maybe put up an outline for orientation after your motivation
- …and signposts at pause points during the talk



Related work

[PMW83]	The seminal paper
[SPZ88]	First use of epimorphisms
[PN93]	Application of epimorphisms to wibblification
[BXX98]	Lacks full abstraction
[XXB99]	Only runs on Sparc, no integration with GUI



Do not present related work

But

- You absolutely must know the related work; respond readily to questions
- Acknowledge co-authors (title slide), and pre-cursors (as you go along)
- Do not disparage the opposition
  - X's very interesting work does Y; I have extended it to do Z

Technical detail (typicaly Greek)

$$\frac{}{\Gamma \vdash k : \tau_k} \qquad \frac{\Gamma \cup \{x : \tau\} \vdash e : \tau'}{\Gamma \vdash \lambda x.e : \tau \to \tau'} \qquad \frac{\Gamma \vdash e_1 : \mathtt{ST}\ \tau^\circ\ \tau \qquad \Gamma \vdash e_2 : \tau \to \mathtt{ST}\ \tau^\circ\ \tau'}{\Gamma \vdash e_1\ \mathtt{>>=}\ e_2 : \mathtt{ST}\ \tau^\circ\ \tau'}$$

$$\frac{\Gamma \vdash e : \tau}{\Gamma \vdash \mathtt{returnST}\ e : \mathtt{ST}\ \tau^\circ\ \tau} \qquad \frac{\Gamma \vdash e : \tau}{\Gamma \vdash \mathtt{newVar}\ e : \mathtt{ST}\ \tau^\circ\ (\mathtt{MutVar}\ \tau^\circ\ \tau)} \qquad \frac{\Gamma \vdash e : \mathtt{MutVar}\ \tau^\circ\ \tau}{\Gamma \vdash \mathtt{readVar}\ e : \mathtt{ST}\ \tau^\circ\ \tau}$$

$$\frac{\Gamma \vdash e_1 : \mathtt{MutVar}\ \tau^\circ\ \tau \qquad \Gamma \vdash e_2 : \tau}{\Gamma \vdash \mathtt{writeVar}\ e_1\ e_2 : \mathtt{ST}\ \tau^\circ\ \mathtt{Unit}} \qquad \frac{}{\Gamma \cup \{x : \forall \alpha_i.\tau\} \vdash x : \tau[\tau_i/\alpha_i]}$$

$$\frac{\Gamma \vdash e : \tau' \to \tau \qquad \Gamma \vdash e' : \tau'}{\Gamma \vdash e\ e' : \tau} \qquad \frac{\Gamma \vdash e : \mathtt{ST}\ \alpha^\circ\ \tau}{\Gamma \vdash \mathtt{runST}\ e : \tau}\ \alpha^\circ \notin FV(\Gamma, \tau)$$

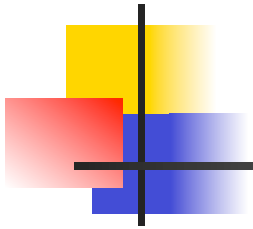
$$\frac{\forall j.\Gamma \cup \{x_i : \tau_i\}_i \vdash e_j : \tau_j \qquad \Gamma \cup \{x_i : \forall \alpha_{j_i}.\tau_i\}_i \vdash e' : \tau'}{\Gamma \vdash \mathtt{let}\ \{x_i = e_i\}_i\ \mathtt{in}\ e' : \tau'}\ \alpha_{j_i} \in FV(\tau_i) - FV(\Gamma)$$

Figure 1. Typing Rules

Omit technical details

- Even though every line is drenched in your blood and sweat, dense clouds of notation will send your audience to sleep

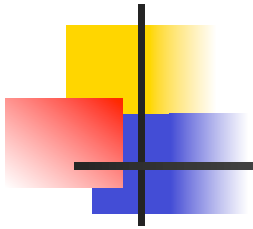
- Present specific aspects only; refer to the paper for the details
- By all means have backup slides to use in response to questions



Do not apologise

- "I didn't have time to prepare this talk properly"
- "My computer broke down, so I don't have the results I expected"
- "I don't have time to tell you about this"
- "I don't feel qualified to address this audience"

Presenting your talk



Write your slides the night before

(not like these☺…or at least, polish it then)

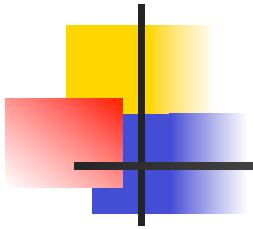
Your talk absolutely must be fresh in your mind

- Ideas will occur to you during the conference, as you obsess on your talk during other people's presentations
- Do not use typeset slides, unless you have a laptop too
- Handwritten slides are fine
  - Use permanent ink
  - Get an eraser: toothpaste does not work

How to present your talk

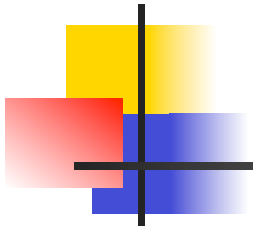
By far the most important thing is to

be enthusiastic



Enthusiasm

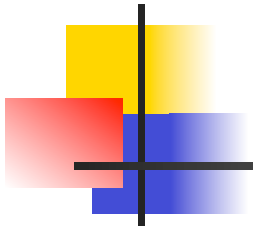
- If you do not seem excited by your idea, why should the audience be?
- It wakes 'em up
- Enthusiasm makes people dramatically more receptive
- It gets you loosened up, breathing, moving around



The jelly effect

If you are anything like me, you will experience apparently-severe pre-talk symptoms

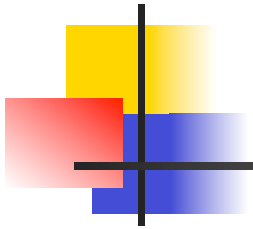
- Inability to breathe
- Inability to stand up (legs give way)
- Inability to operate brain



What to do about it

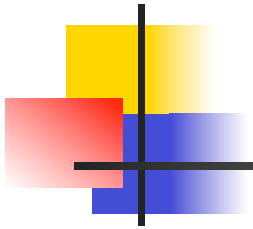
- Deep breathing during previous talk
- Yoga and Tai Chi are not bogus either
- *Script your first few sentences precisely*
  (=> no brain required)
- Move around a lot, use large gestures, wave your arms, stand on chairs
- Go to the loo first

- You are not a wimp. Everyone feels this way.



Being seen, being heard

- Point at the screen, not at the overhead projector
- Speak to someone at the back of the room, even if you have a microphone on
- Make eye contact; identify a nodder (you for example), and speak to him or her (better still, more than one)
- Watch audience for questions…



Questions

- Questions are not a problem
- Questions are a golden golden golden opportunity to connect with your audience
- Specifically encourage questions during your talk: pause briefly now and then, ask for questions
- Be prepared to truncate your talk if you run out of time. Better to connect, and not to present all your material

Presenting your slides

A very annoying technique

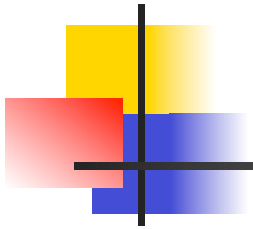
- is to reveal
- your points
- one
- by one
- by one, unless…
- there is a punch line

Presenting your slides

Use animation effects

very very very very very very very

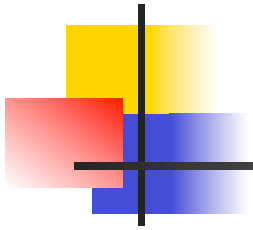
sparingly



Finishing

Absolutely without fail, finish on time

- Audiences get restive and essentially stop listening when your time is up. Continuing is very counter productive
- Simply truncate and conclude
- Do not say "would you like me to go on?" (it's hard to say "no thanks")



There is hope

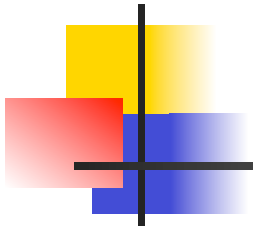
The general standard is so low that you don't have to be outstanding to stand out

You will attend 50x as many talks as you give. Watch other people's talks intelligently, and pick up ideas for what to do and what to avoid.

5. How to write a great research grant proposal

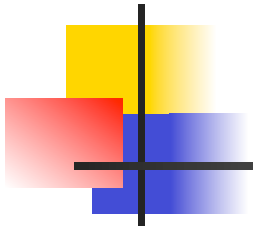
Simon Peyton Jones,
Microsoft Research, Cambridge

with Alan Bundy, Edinburgh University



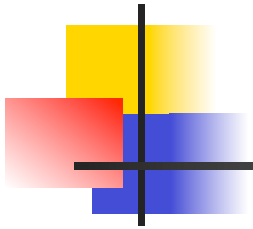
Grants are important

- Research grants are the dominant way for academic researchers to get resources to focus on research
- **INVARIANT**: there is never enough money



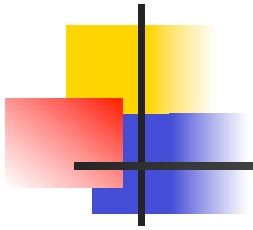
The state of play

- Even a strong proposal is in a lottery, but a weak one is certainly dead
- Many research proposals are weak
- Most weak proposals could be improved quite easily



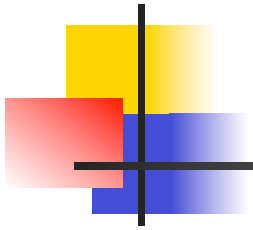
Audience

- **With luck**, your proposal will be read **carefully** by one or two **experts**. You must convince them.
- But it will **certainly** be read **superficially** by **non-experts**… and *they* will be the panel members. You absolutely must convince them too.
- Some influential readers will be non-experts, and will give you **one minute maximum**.



The vague proposal

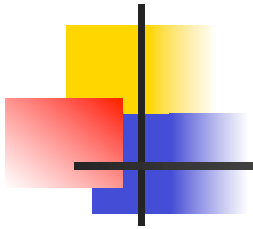
1. I want to work on better type systems for functional programming languages
2. **Give me the money**



The vague proposal

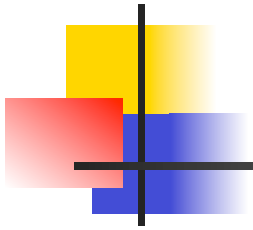
1. I want to work on better type systems for functional programming languages
2. **Give me the money**

You absolutely must identify the problem you are going to tackle



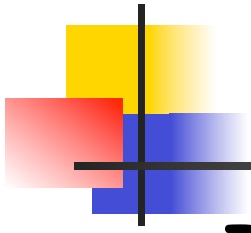
Identifying the problem

- What is the problem?
- Is it an **interesting** problem? That is, is it research at all?
- Is it an **important** problem? That is, would anyone care if you solved it? (jargon: "impact")
- Having a "customer" helps: someone who wants you to solve the problem



The aspirational proposal

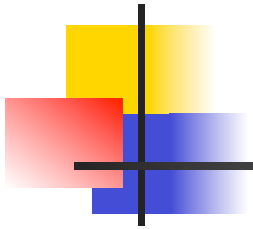
1. I want to solve the problem of avoiding all deadlocks and race conditions in all concurrent and distributed programs
2. **Give me the money**



The aspirational proposal

1. I want to solve the problem of avoiding deadlocks and race conditions in concurrent and distributed programs
2. **Give me the money**

- **It is easy to identify an impressive mountain**
- **But that is not enough! You must convince your reader that you stand some chance of climbing the mountain**



Climbing the mountain

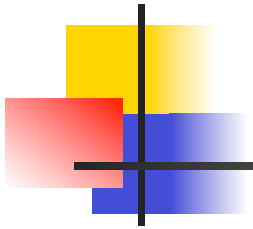
Two sorts of evidence

1. You absolutely must say what is the

   idea

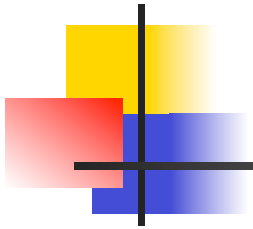
   that you are bringing to the proposal.

2. Explain modestly but firmly why **you** are ideally equipped to carry out this work. (NB: not enough without (1))



1. Your idea

- Give real technical "**meat**", so an expert reader could (without reading your doubtless-excellent papers) have some idea of what the idea is
- Many, many grant proposals have impressive sounding words, but lack almost all technical content. Reject!

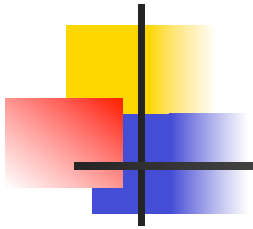


1. Your idea

Offer **objective evidence** that it's a **promising** idea:

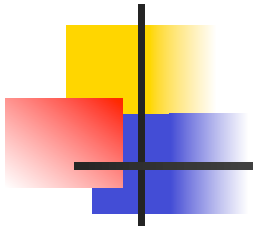
- Results of preliminary work
- Prototypes
- Publications
- Applications

Strike a balance: you don't want the reader to think "they've already solved the problem".



2. Blowing your own trumpet

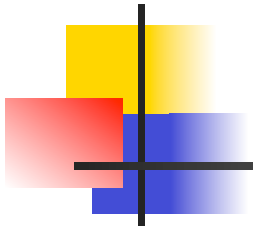
- Grants fund **people**
- Most researchers are far too modest. "It has been shown that …[4]", when [4] is you own work!
- Use the first person: "I did this", "We did that".
- Do not rely only on the boring "track record" section



2. Blowing your own trumpet

Express value judgements using strong, but defensible, statements: pretend that you are a well-informed but unbiased expert

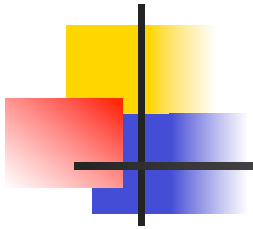
- "We were the first to …"
- "Out 1998 POPL paper has proved very influential…"
- "We are recognised as world leaders in functional programming"
- Google Scholar, Citeseer citation count/ impact
- H-factor or Erdos number☺



2. Blowing your own trumpet

Choose your area...

- "We are recognised as world leaders in
  - functional programming
  - Haskell
  - Haskell's type system
  - functional dependencies in Haskell's type system
  - sub-variant X of variant Y of functional dependencies in Haskell's type system"



Your message

We are ideally placed to do this timely research because

- We have an idea
- Our preliminary work shows that it's a promising idea
- We are the best in our field

The arrogant proposal

1. I am an Important and Famous Researcher. I have lots of PhD students. I have lots of papers.
2. **Give me the money**

- Proposals like this do sometimes get funded. But they shouldn't.
- Your proposal should, **all by itself**, justify your grant

The I'll-work-on-it proposal

1. Here is a (well-formulated, important) problem
2. Here is a promising idea (…evidence)
3. We're a great team (…evidence)
4. We'll work on it
5. **Give us the money**

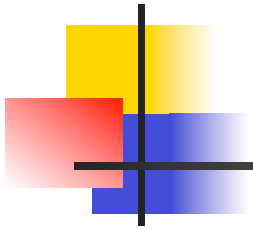
The I'll-work-on-it proposal

1. Here is a (well-formulated, important) problem
2. Here is a promising idea
3. We're a world-class team
4. We'll work on it
5. **Give us the money**

The key question

How would a reviewer know if your research had succeeded?

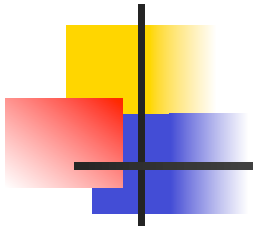
Jargon: "aims, objectives"



Suspicious phrases

- "Gain insight into…"
- "Develop the theory of…"
- "Study…"

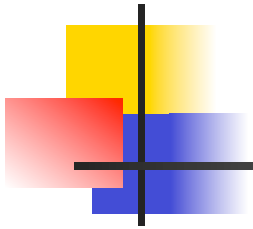
The trouble with all of these is that there is no way to distinguish abject failure from stunning success.



Good phrases

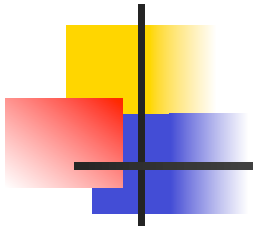
- "We will build an analyser that will analyse our 200k line C program in less than an hour"
- "We will build a prototype walkabout information-access system, and try it out with three consultants in hospital Y"

The most convincing success criteria involve those "customers" again



Related work

- **Goal 1**: demonstrate that you totally know the field. Appearing ignorant of relevant related work is certain death.
- **Goal 2**: a spring-board for describing your promising idea
- But that is all! Do not spend too many words on comparative discussion. The experts will know it; the non-experts won't care.

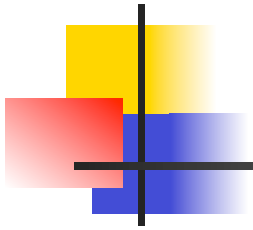


Methodology/plans

- "Methodology", or describing your step-by-step plans, is usually over-stressed in my view.
- Concentrate on (a) your idea, and (b) your aims/objectives/success criteria. Then the "methodology" part writes itself.

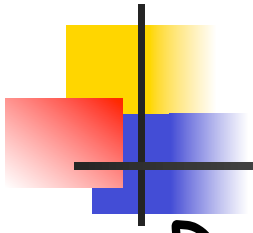
The ideal proposal

1. Here is a well-defined problem
2. It's an important problem (evidence…)
3. We have a promising idea (evidence…)
4. We are a world-class team (evidence…)
5. Here is what we hope to achieve
6. Here is how we plan to build on our idea to achieve it
7. **Give us the money. Please.**



One page, please

- Start with a **one-page summary**, that tells the whole story (previous slide)
- Remember: most of your readers will only read this page
- **NO BOILERPLATE**: "The XYZ institute has a vigorous research programme in many important fields...". This page is worth **10x** the other pages. Every word is precious.
- See section on reading and writing for how much time someone spends and how many people ***stop at this point***



Know your agency

- Read the call for proposals
- Try to understand what the motivation of the agency (or company) is
- Understand their criteria, and write your proposal to address them
- But do not prostitute your research. Write a proposal for good research that you are genuinely excited about.
- Do not exceed the page limit

Know your agency

- Find a reason to telephone (not email) the program manager. S/he is a Human Being, and is constantly on the lookout for original research.
- Build your relationship. Invite them to visit your institute. Offer to help as a reviewer. Ask what you can do that would help them. Do not begin by making demands (everyone else does)

Help each other

Ask others to read your proposal critically
Revise, and ask someone else
Repeat

- Cheap: what someone thinks after a 10-minute read is Really Really Important
- Informative: after reading 20 proposals by others, you'll write better ones yourself. You know what is good and bad now. Much better proposals.
- Effective: dramatic increases in quality. There is just no excuse for not doing this.

Attitude

- To every unfair, unjustified, and ill-informed criticism from your reader, respond *"That's very interesting… here is what I intended to say… how could I rephrase it so that you would have understood that"?*
- Better get criticised by your friendly colleagues than by panel member at the meeting.
- Much easier do face to face than by email

Good news

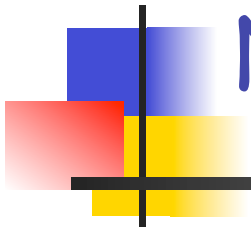
The general standard of research proposals is low

So it is not hard to shine

Although, sadly, that still does not guarantee a grant.

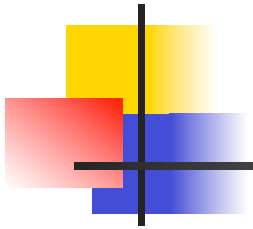
Good luck!

http://research.microsoft.com/~simonpj/papers/Proposal.html



6. Re-heating Cold Topics in Networks

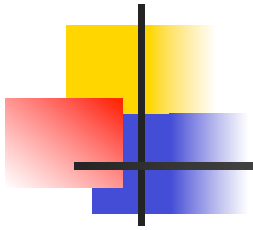
Jon Crowcroft, Cambridge



Hot v. Cold

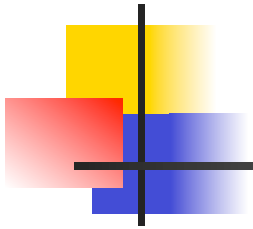
Research goes in cycles - possibly Carnot Cycles

- Topics become hot again!
- Initially, even, controversial
- Cold Topic gets re-heated.
- See earlier ideas
  - Combine two cold topics
  - Stretch one dimension to new extreme
  - Combine a cold and a hot topic



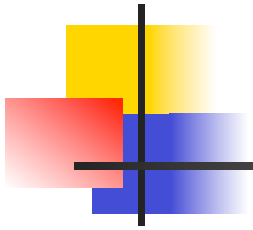
Some examples of cold topics

- [DHT and Structured P2P]
  - Mobile, social networks and DHTs
  - Label social graph, and map from content key to label (e.g. for address book mutual backup)
- [Internet Coordinate Systems]
  - Decentralised ICS for VANETs
- [Faster packet classification]
  - Lower power consumption? Greening Internet is now good
- [BGP] The Border Gateway Protocol
  - Application layer policy routing (skype



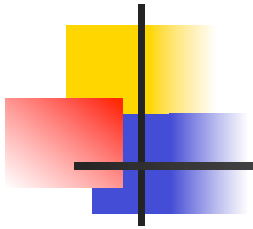
More cold topics

- DDoS
  - Botnets for good. Test traffic? Light control plane
- Spam
  - Social net based rate limiting
- Overlays
  - Evolution (continuous migration of overlay service to native layer)



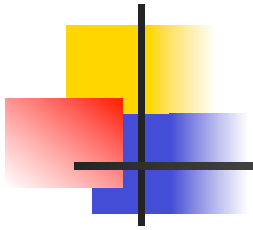
Even more cold topics

- TINA
  - TINA for DTNs (in F# for cell phones)
- TCP+AQM
  - Validation and reprodceability of research
- Multicast
  - Software patch carousel (for unreliable cell phone OS☺?
- Newarch
  - Packet swarms



Yet more cold topics

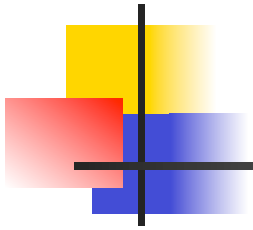
- Self similarity
  - Power laws in human mobility
- MANETs
  - Perhaps with low power and WILD 802.11
- Self Organising WSNs
  - Self organising component software?
- Small World Networks
  - Ask epidemiologists



Now you know what I do like…

For now :-)

Your PhD topic may have been odd in Year 1. By year 3,4,5 it might become :-).



At the end of your career…

- Do you want 100 PhD students
- 1000 papers
- 100 patents
- 100M euros of projects
- 1M lines of code
- 10 books…

- Metrics are very very annoying and reductionist
- Van Jacobson wrote the highest cited paper in CS in 1988 - he only has about 10 papers in all:-) His algorithm runs in>1 billion computers.