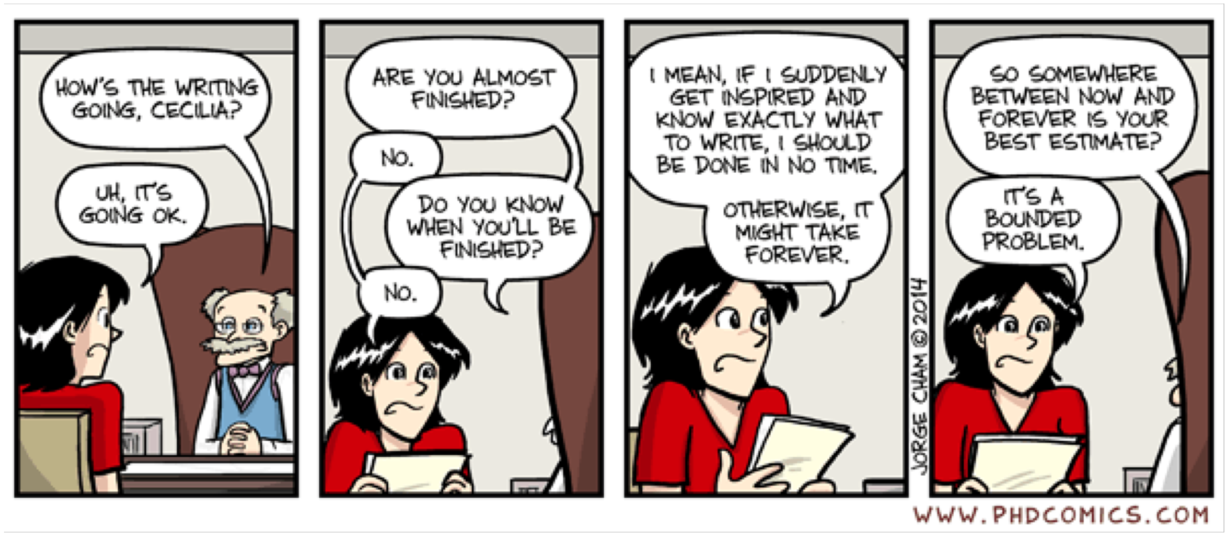
How to write a great research paper

To do …

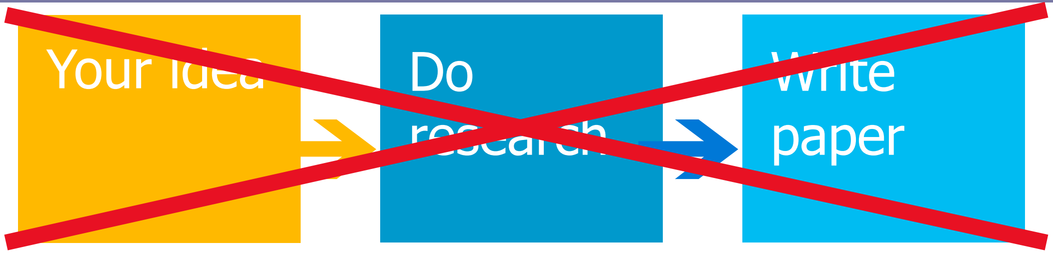
- Seven simple, actionable suggestions that will make your papers better.


HOW'S THE WRITING GOING, CECILIA?
UH, IT'S GOING OK.
ARE YOU ALMOST FINISHED?
NO.
DO YOU KNOW WHEN YOU'LL BE FINISHED?
NO.
I MEAN, IF I SUDDENLY GET INSPIRED AND KNOW EXACTLY WHAT TO WRITE, I SHOULD BE DONE IN NO TIME.
OTHERWISE, IT MIGHT TAKE FOREVER.
SO SOMEWHERE BETWEEN NOW AND FOREVER IS YOUR BEST ESTIMATE?
IT'S A BOUNDED PROBLEM.
JORGE CHAM © 2014
WWW.PHDCOMICS.COM

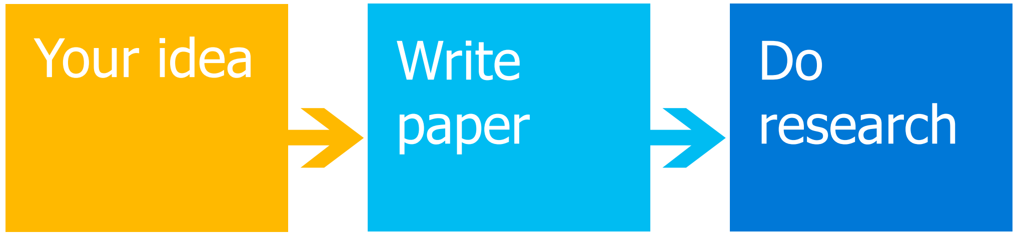
Based on slides by Simon Peyton Jones, MSR Cambridge

1. Don’t wait: write

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

Writing as a mechanism for doing research
(not just for reporting it)

2. Identify your key idea

Your goal: to convey a useful and re-usable idea

- You want 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

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

- 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

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

The idea

Idea:

A re-usable insight, useful to the reader

- Your paper should have just one “ping”: one clear, sharp idea
- 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

The idea

- Many papers contain good ideas, but do not distil what they are.
- 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.”

3. Tell a story

Your narrative flow

Imagine you are explaining at a whiteboard

- 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)

4. Nail your contributions to the mast

The introduction (1 page)

- Describe the problem (forward)
- State your contributions (summary)

...and that is all

ONE PAGE!

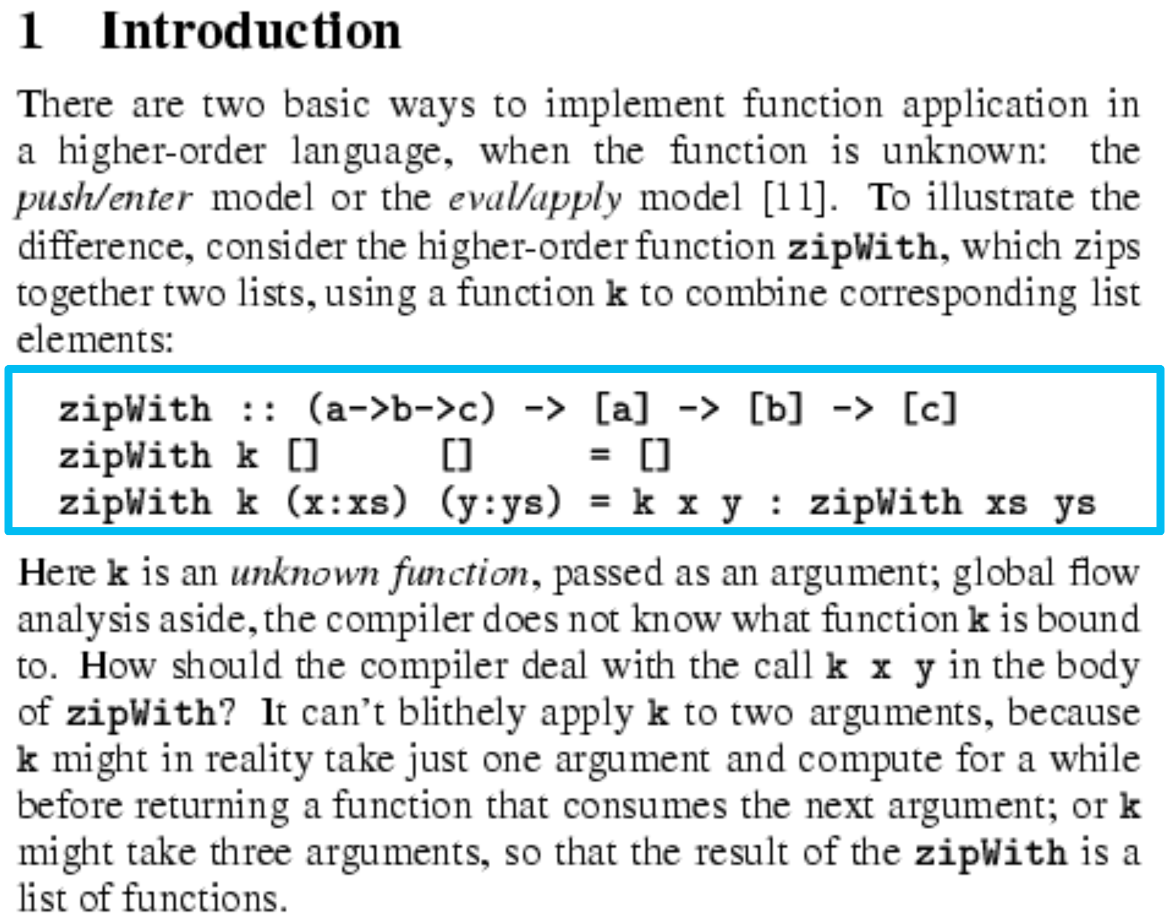
Describe the problem

Forward: The situation before the work was done

- [Context] – Why the need is so pressing or important
- Need – Why something needs to be done at all
- Task – What was done to address it
  - Connector - a simple “therefore” or desire part of the need
  - Active voice, first person, past or present perfect (“We measured”)
- Object – What the present document does/covers
  - Focus on the document which is atemporal (so present tense)

Describe the problem

Use an example to introduce 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.

Molehills not mountains

Example: "Computer programs often have bugs. It is very important to eliminate these bugs [1,2]. Many researchers have tried [3,4,5,6]. It really is very important."

Yawn!

Example: "Consider this program, which has an interesting bug. <brief description>. We will show an automatic technique for identifying and removing such bugs"

Cool!

Summary /conclusion

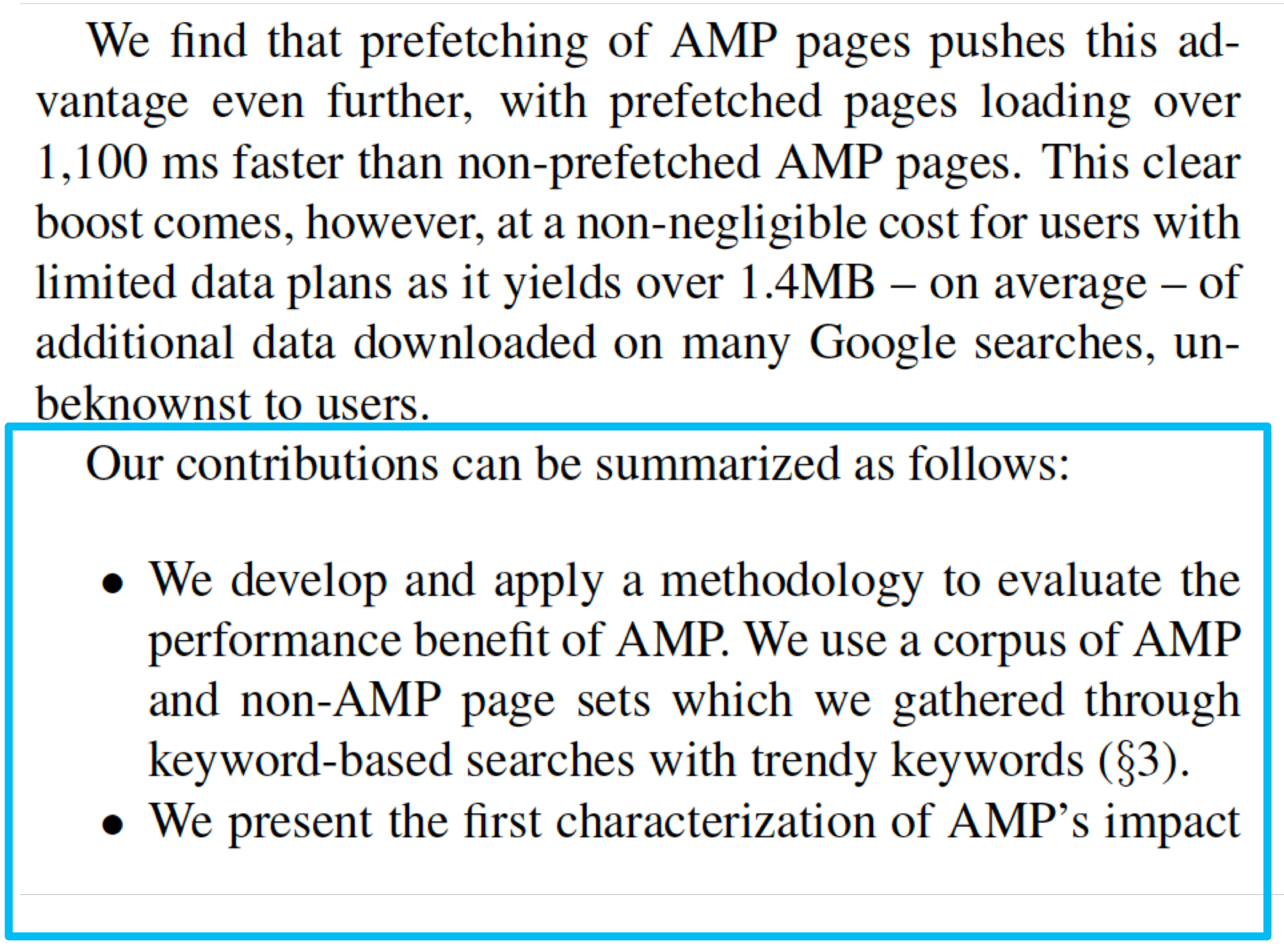
Summary: The situation after

- Findings – Main results (what we found doing the task)
- Conclusions – What the findings mean to the audience
- [Perspectives] – What the future holds, beyond this work

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



We find that prefetching of AMP pages pushes this advantage even further, with prefetched pages loading over 1,100 ms faster than non-prefetched AMP pages. This clear boost comes, however, at a non-negligible cost for users with limited data plans as it yields over 1.4MB – on average – of additional data downloaded on many Google searches, unbeknownst to users.

Our contributions can be summarized as follows:

- We develop and apply a methodology to evaluate the performance benefit of AMP. We use a corpus of AMP and non-AMP page sets which we gathered through keyword-based searches with trendy keywords (§3).
- We present the first characterization of AMP's impact

Bulleted
list of contributions

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

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.

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

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.

5. Related work: later

Structure

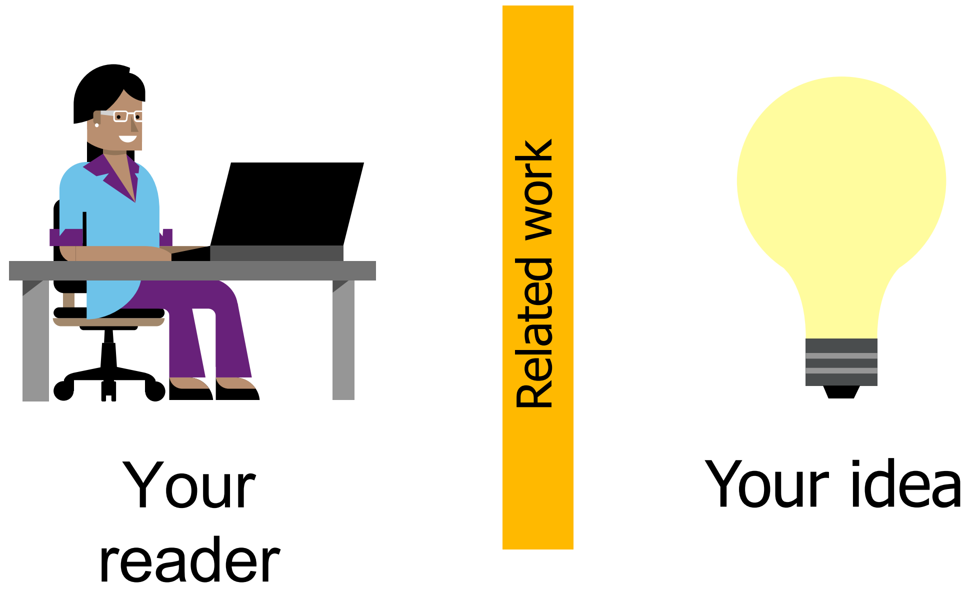
- Abstract (4 sentences)
- Introduction (1 page)
- Related work
- The problem (1 page)
- My idea (2 pages)
- The details (5 pages)
- Conclusions and further work (0.5 pages)

NO!

YES!

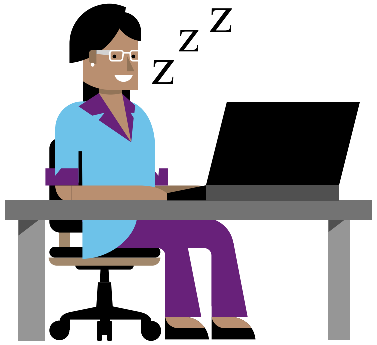
- Abstract (4 sentences)
- Introduction (1 page)
- The problem (1 page)
- My idea (2 pages)
- The details (5 pages)
- Related work (1-2 pages)
- Conclusions and further work (0.5 pages)

No related work yet!



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 (highly compressed) description of various technical tradeoffs is absolutely incomprehensible
- Problem 2: describing alternative approaches gets between the reader and your idea

Credit

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

Truth: Credit is not like money

- Warmly acknowledge people who have helped you
- Be generous to the competition.
- Acknowledge weaknesses in your approach

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

6. Put your readers first

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)

Structure

3. The idea

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

- Sounds impressive...but
- Sends readers to sleep, and/or makes them feel stupid

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

Conveying the intuition

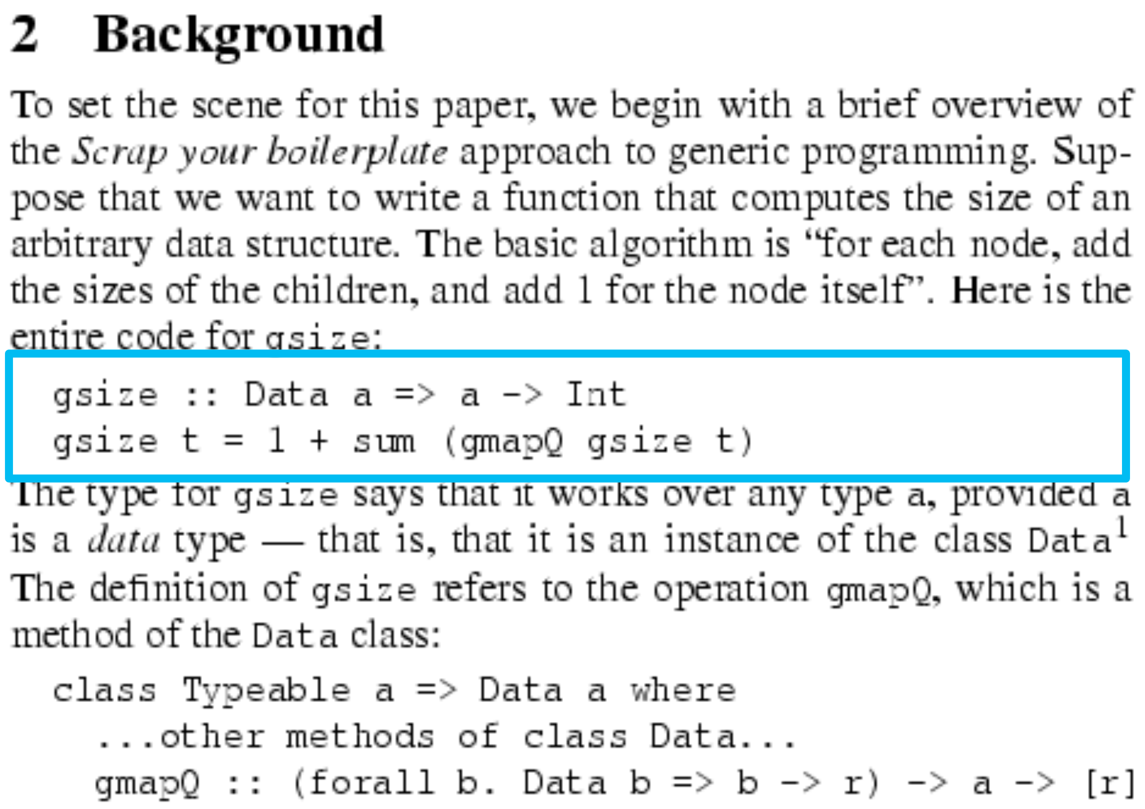
Introduce the problem, and your idea, using EXAMPLES and only then present the general case

- Remember: explain it as if you were speaking to someone using a whiteboard

Using examples

The Simon PJ question: is there any typewriter font?

Example right away



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]
```

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.

7. Listen to your readers

Getting help

- 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”.)

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

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".
- INSTEAD: fix the paper so that X is apparent even to the stupidest reader.
- Thank them warmly. They have given up their time for you.

Summary

1. Don’t wait: write
2. Identify your key idea
3. Tell a story
4. Nail your contributions
5. Related work: later
6. Put your readers first (examples)
7. Listen to your readers

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

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

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