

Research Methodology: How to write a great research paper

Simon Peyton Jones (Microsoft Research)

Revised by Chakchai So-In, Ph.D.

Department of Computer Science
Faculty of Science, Khon Kaen University
123 Mitaparb Rd., Naimaung,
Maung, Khon Kaen, 40002 Thailand


chakso@kku.ac.th

http://web.kku.ac.th/chakso/322793_Spring11/

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

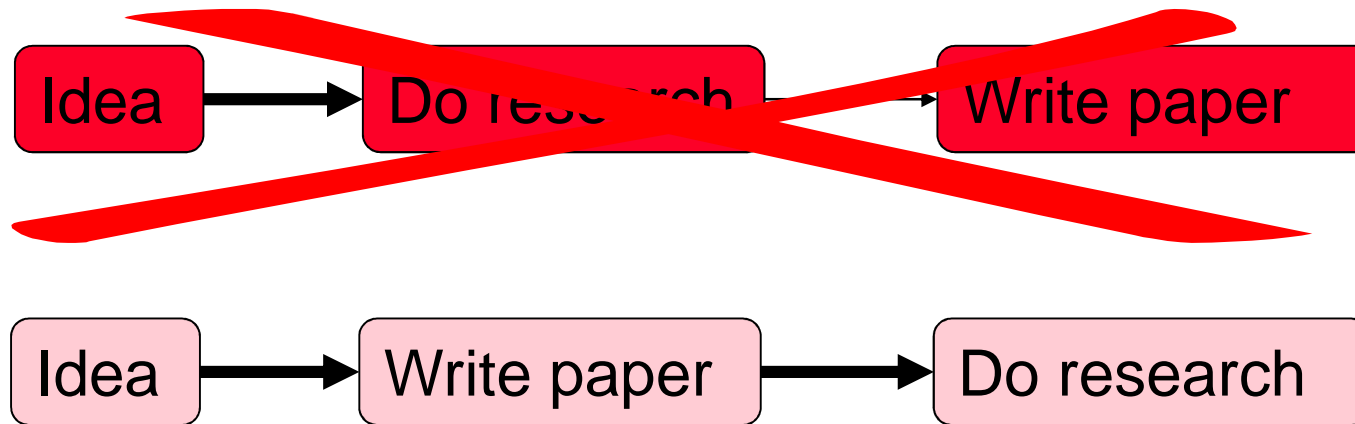
Increasing importance



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

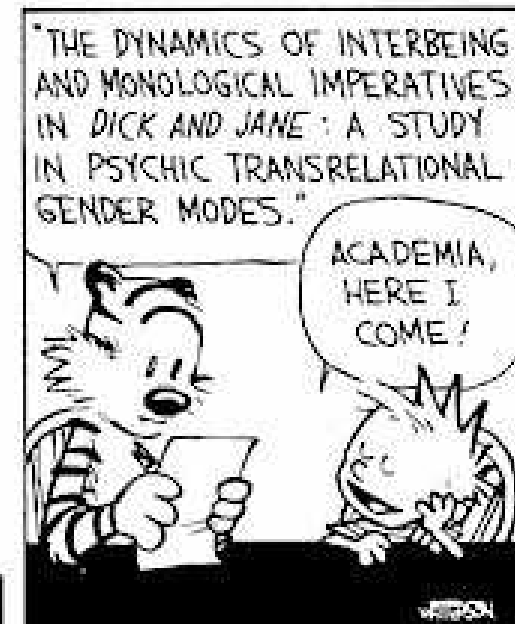
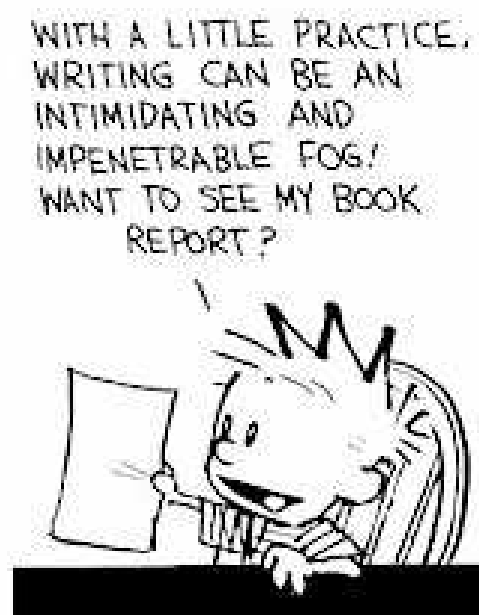
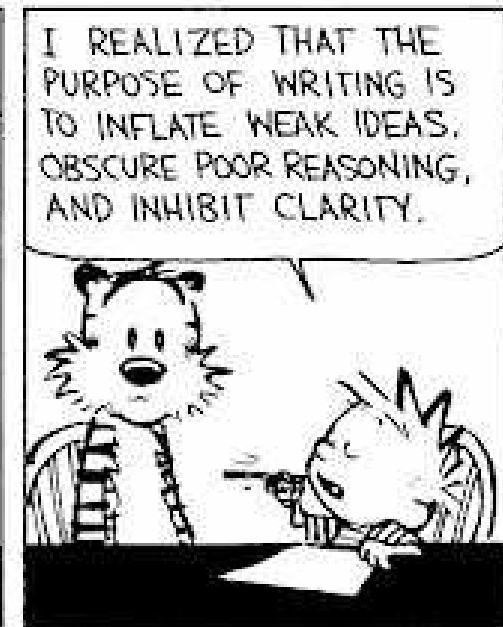
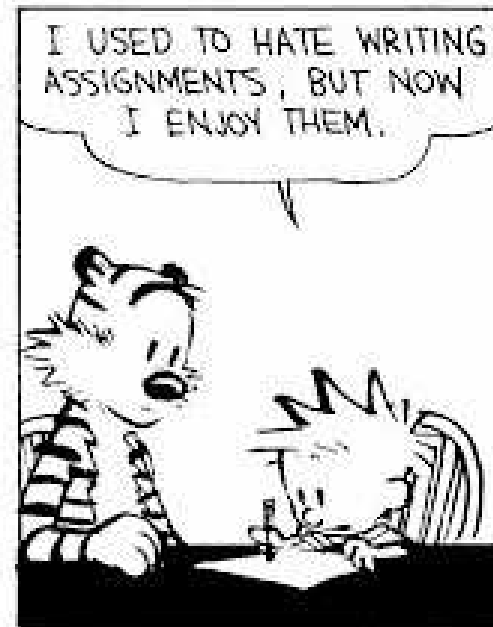
- ❑ Writing the paper is how you develop the idea in the first place
- ❑ It usually turns out to be more interesting and challenging than it seemed at first

The purpose of your paper

Why bother?

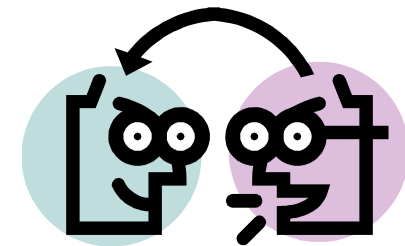
Fallacy

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

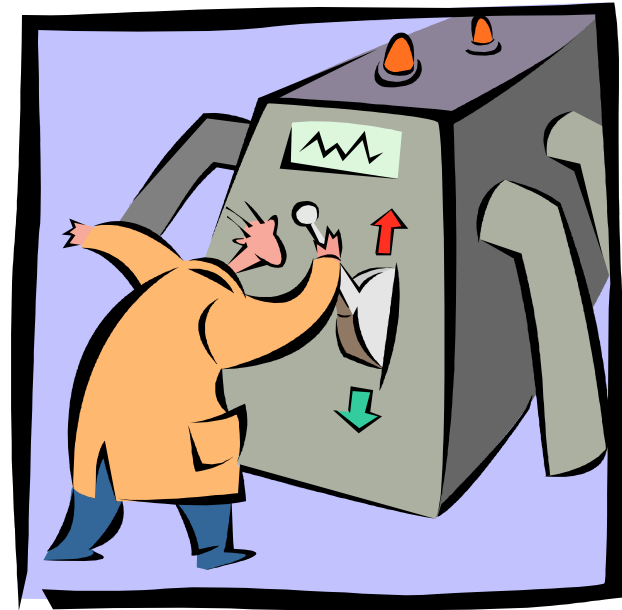
- ❑ Figure out what your idea is
- ❑ Make certain that the reader is in no doubt what the idea is. Be 100% explicit:
 - “The main idea of this paper is....”
 - “In this section we present the main contributions of the paper.”
- ❑ Many papers contain good ideas, but do not distil what they are.

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

The purpose of your paper is not...

To describe the
WizWoz system



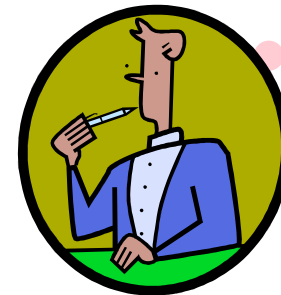
- ❑ Your reader does not have a WizWoz
- ❑ She is primarily interested in re-usable brain-stuff, not executable artefacts

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

I wish I knew how to solve that!

I see how that works. Ingenious!



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)

Title and Keywords

- ❑ Title
 - Based on Overall Paper; Searchable
- ❑ Key Search Words
 - All words **MUST** be in abstract; Acronyms and full names

The “*abstract*”

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

Examples

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)
- ❑ References (0.25 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

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 (cont.)

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!

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.
- ❑ Note: this *depends* on the structure of the paper (conference requirements)

Structure

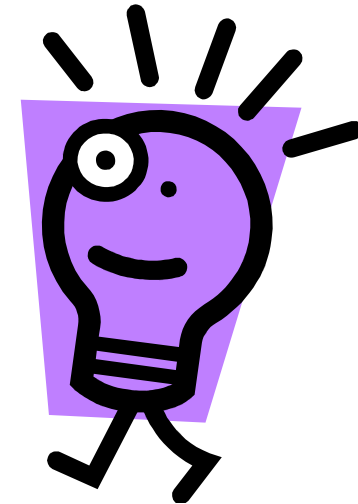
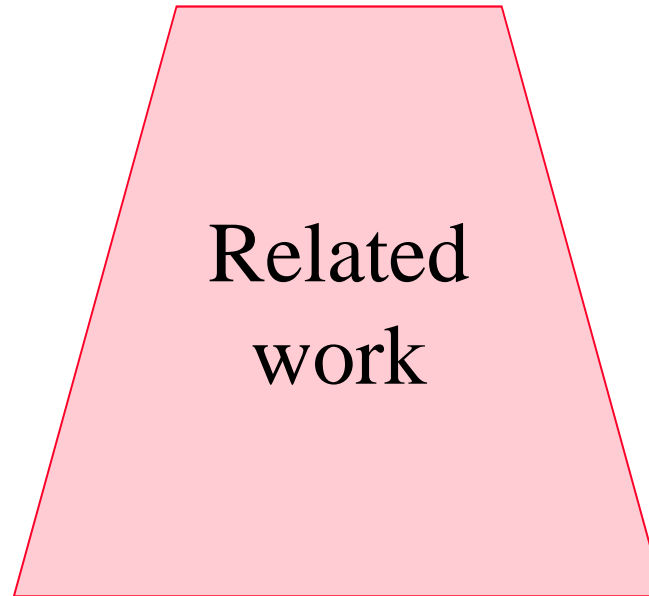
- ❑ Abstract (4 sentences)
- ❑ Introduction (1 page)
- ~~❑ Related work~~
- ❑ The problem (1 page)
- ❑ My idea (2 pages)
- ❑ The details (5 pages)
- ❑ Related work (1-2 pages)
- ❑ Conclusions and further work (0.5 pages)
- ❑ References (0.25 pages)

Note: this depends on the structure of the paper (conference requirements)

No related work yet!



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! (cont.)

- ❑ **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

I feel
stupid



I feel
tired

Note: this depends on the structure of the paper (conference requirement)

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)
- ❑ References (0.25 pages)

Presenting the idea

3. The idea

Consider a bifurcated 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 epimodulus 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 (cont.)

- ❑ 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

Results

- ❑ Describe the platform, processor, OS, language, compiler, compiler optimizations
- ❑ Describe the test suite, which s/b legit! You can't just make up the test programs!!
- ❑ Use tables: describe each row & column
- ❑ Sometimes a graph is better than a table

graphs are great!

“A picture is worth...”

Test case	ATT	Tl	EOP	None
encrypt	11.62	0.96	0.53	0.53
php2cpp	6.66	1.24	0.77	0.77
ft	25.84	3.32	2.37	2.33
graphdraw	54.22	7.55	5.43	5.40
ep matrix	175.56	35.04	29.42	29.29
vkey	133.22	19.01	15.53	15.53

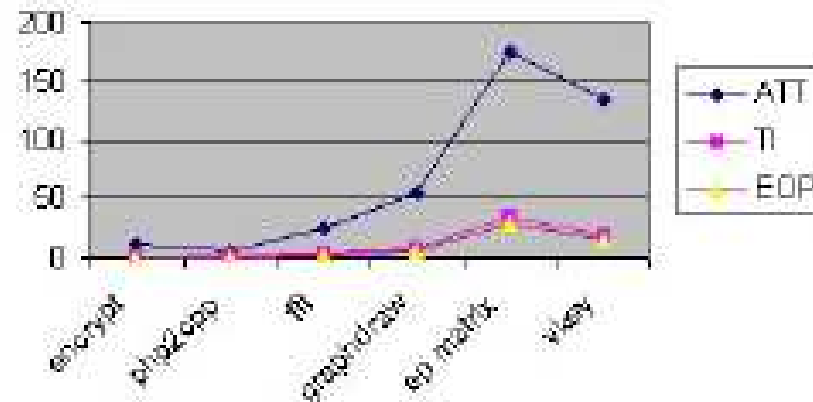
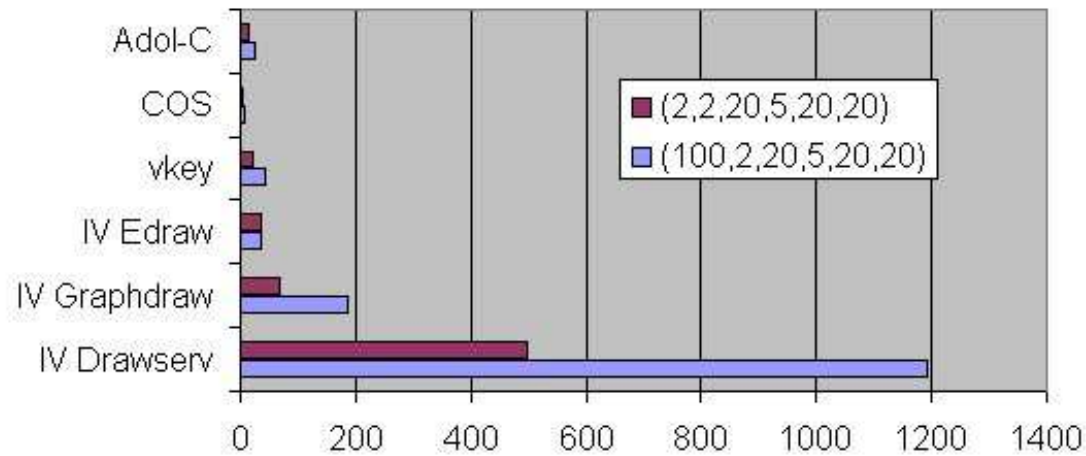


Figure 8. Efficiency results.



Test case	#edges removed (2,2,20,5,20,20)	#edges removed (100,2,20,5,20,20)
Adol-C	13	26
COS	2	6
vkey	20	44
IV Edraw	35	37
IV graphdraw	69	185
IV drawserv	498	1191

Figure 11. *Number of edges removed.* The table lists the number of edges removed to break cycles using two cost models that differ by the weight assigned to inheritance edges. The bar graph highlights the difference in the two models.

Impact of Results

- ❑ How many times did you perform each experiment?
- ❑ Validity of results
 - Any weaknesses of the test suite
 - Anything “hokey” about the approach
- ❑ Threats to generalize
 - Can you really generalize: why/why not
 - Is it **automatable** or **automated**?!

Most important!

- ❑ Do not claim more than you did
- ❑ Do not generalize from one study or result
- ❑ Do not claim that because it worked well on a few test cases that it will work well on all test cases, all platforms and for all inputs!

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)
- ❑ References (0.25 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)
- ❑ References (0.25 pages)

Conclusions and Future Work

- ❑ Be brief.
 - One or more sentences about each issue;
 - Key lessons
 - Say what you can't do (limitation); and want to do

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)
- ❑ **References** (0.25 pages)

References

- ❑ Style of References (See [IEEE Journal Format](#))
 - Author(s), “Title,” Source, date, pages, [URL](#)
 - The URL should show up.
 - All references should be annotated and have links.
- ❑ Order in the most important first and indicate so
- ❑ In the text point to the end
- ❑ Remove references that are useless.
- ❑ [Number] notation according to order in referred text
- ❑ Exception: Standards, company documents, RFCs.

Literature Search

- ❑ Finding references: Use Google advanced search options
 - Location 802.11 +filetype:pdf +site:.com
 - See advanced search in Google:
http://www.google.com/advanced_search?hl=en
 - <http://ieeexplore.ieee.org/Xplore/guesthome.jsp>
 - <http://academic.research.microsoft.com>
- ❑ Conduct searches in two phases. In the first phase, use the title words of your project. After reading these, conduct another more comprehensive search.
- ❑ Remove articles that are not useful
- ❑ No limit to the number of references
- ❑ Follow the references in references

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 (cont.)

- ❑ 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

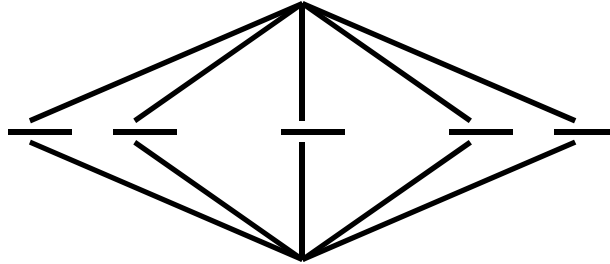
Basic Stuff

- ❑ 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

Writing Style

- ❑ Readers want to get to the information fast. Keep the nonessential stuff at the end.
- ❑ Check thoroughly for grammar and spelling.
- ❑ Avoid excessive use of abbreviations.
- ❑ Be consistent in case and usage: MOBILE, Mobile, mobile

Diamond Writing Style



- ❑ Each paper should start with an introduction and end with a summary.
- ❑ Each section should start with a short introduction and end with a summary with a lead in to the next section. The same applies to subsections.
- ❑ All subsections should be of comparable length.
- ❑ Add an appendix with all abbreviations
- ❑ Add a list or discussion of related products

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 (cont.)

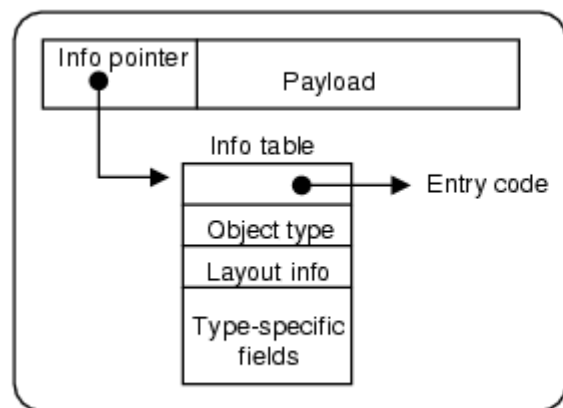


Figure 3. A heap object

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 $CON(C a_1 \dots a_n)$ would be represented by an object whose info pointer represented the constructor C and whose payload is the arguments $a_1 \dots a_n$.

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

It can be seen that...

34 tests were run

These properties were
thought desirable

It might be thought that
this would be a type error

YES

We can see that...

We ran 34 tests

We wanted to retain these
properties

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

The object under study was displaced horizontally

On an annual basis

Endeavour to ascertain

It could be considered that the speed of storage reclamation left something to be desired

YES

The ball moved sideways

Yearly

Find out

The garbage collector was really slow

Use present tense

- Present tense is stronger than future tense

WEAK: In this paper, we will show...

STRONG: In this paper, we show that ...

Use present tense

- ❑ Past tense degrades into “**diary writing**”

BAD: In this work we wanted to ...

GOOD: The goals of our work are to ...

Don't change tense

BAD: In this chapter, we have described what happens when we do the wrong thing. We examined the behavior and determine that they are correct.

BAD: The analysis reported in the preceding section will show that Nick can differentiate shod from shoddy.

Be consistent: call a spade a spade!

- ❑ Can't change terminology, even for a good reason, w/out explanation

remote proxy vs proxy
data structure vs structure

Semicolon

- ❑ Connects two sentences that are closely related to each other
- ❑ Use a semicolon when what follows constitutes a complete sentence
- ❑ When what follows is a fragment, you must use a comma or an em dash

Commas

- ❑ Commas provide guidance to your reader about how to parse your sentence!
- ❑ Place them wherever a speaker should pause

UGLY: Greg was worried; however he remained calm.

GOOD: Brendan was hungry; however, he remained calm.

OKAY: Lyn and Richard were still puzzled, however many times they reread the directions for assembling the stepper climber; however, they remained calm.

Colon

- ❑ The colon signifies that what follows it expands on or explains what precedes it: this sentence is an example.

Lyn could tell that Red had been out hunting again: There were three mice neatly laid out on the upstairs rug.

- ❑ Frequently a period or an em dash will also work
- ❑ Use at the end of a sentence, followed by a list

GOOD: This talk does not assume that you know the basics: how to form a sentence, how to use words and how to laugh at your mistakes.

Common Mistakes

- ❑ No Figures/ No Tables/ No Keyword
- ❑ Figure/equations fonts too large/too small
- ❑ Figures with no title or number or reference
- ❑ Figures/tables overflowing the margins
- ❑ References with no annotation
- ❑ References not cited/ Incorrect reference style
- ❑ Key pieces of information without references
- ❑ Tables without references
- ❑ Papers too short/too long
- ❑ No comparison of different alternatives
- ❑ No Acronyms

Other (need to check before submission)

- ❑ Look for special characters
- ❑ American or British English (pick one):
 - E.g., Signaling, Synchronization
- ❑ Check for continuity
- ❑ Check coherent and organization (content flow)
- ❑ Break long paragraphs.
- ❑ The paper should be 4-12 pages long (A4 single-space)
- ❑ If you copy any figures, give reference and credit
- ❑ Use the template supplied (See [IEEE Journal Format](#))
- ❑ **Google Translator** (DO NOT USE or USE very carefully)

Acronyms

- ❑ Search the text
- ❑ Define on the first time you use
 - E.g., Multiple Input Multiple Output (MIMO).....
MIMO.... MIMO
- ❑ Avoid multiple uses if used less than 5 times.
- ❑ Exception: Commonly used acronyms, e.g., CPU, I/O,
...

Figures/Tables

- ❑ All figures should be numbered 1, 2, ...
- ❑ All tables should be numbered 1, 2, ...
- ❑ All figures should have a title below the figure
- ❑ All tables should have a title above the table
- ❑ All figures/tables should be referenced in the text and explained.
- ❑ Should be placed close to their references.
- ❑ To prepare figures use blank slides in PowerPoint and save them as .emf files. Use Microsoft image editor to crop the figures for correct white space around them.

Editorial (Review Process)

- ❑ Check all acronyms. All acronyms should be defined on first use.
- ❑ Check capitalization. No unnecessary capitalization. Headers are usually capitalized.
- ❑ Spell Check entire document.
- ❑ *k* in kilo is lower case. *kbps* not Kbps.
- ❑ Leave a space between numbers and units, e.g., 15 km not 15km.
- ❑ Remember to submit copies of unusual references (not available in the library or the Internet) with the **final** paper.

Summary

If you remember nothing else:

- ❑ **Problem (necessity)?**
- ❑ **Identify your key idea**
- ❑ **Make your Contributions explicit**
- ❑ **Use examples**
- ❑ **Convincing readers by performing performance (statistical) analysis (graph + table?)**

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>

Thank you and Question?

