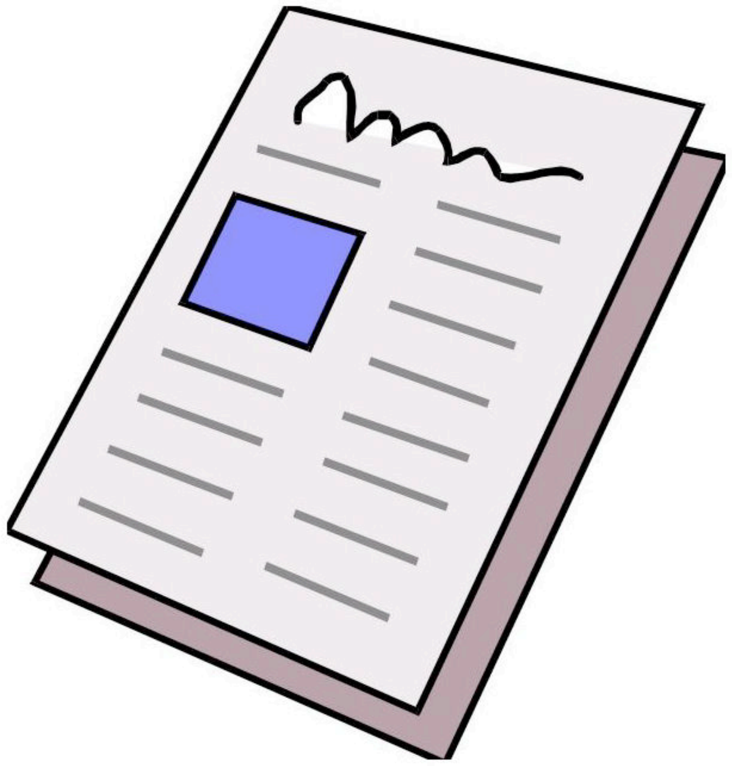
DTU



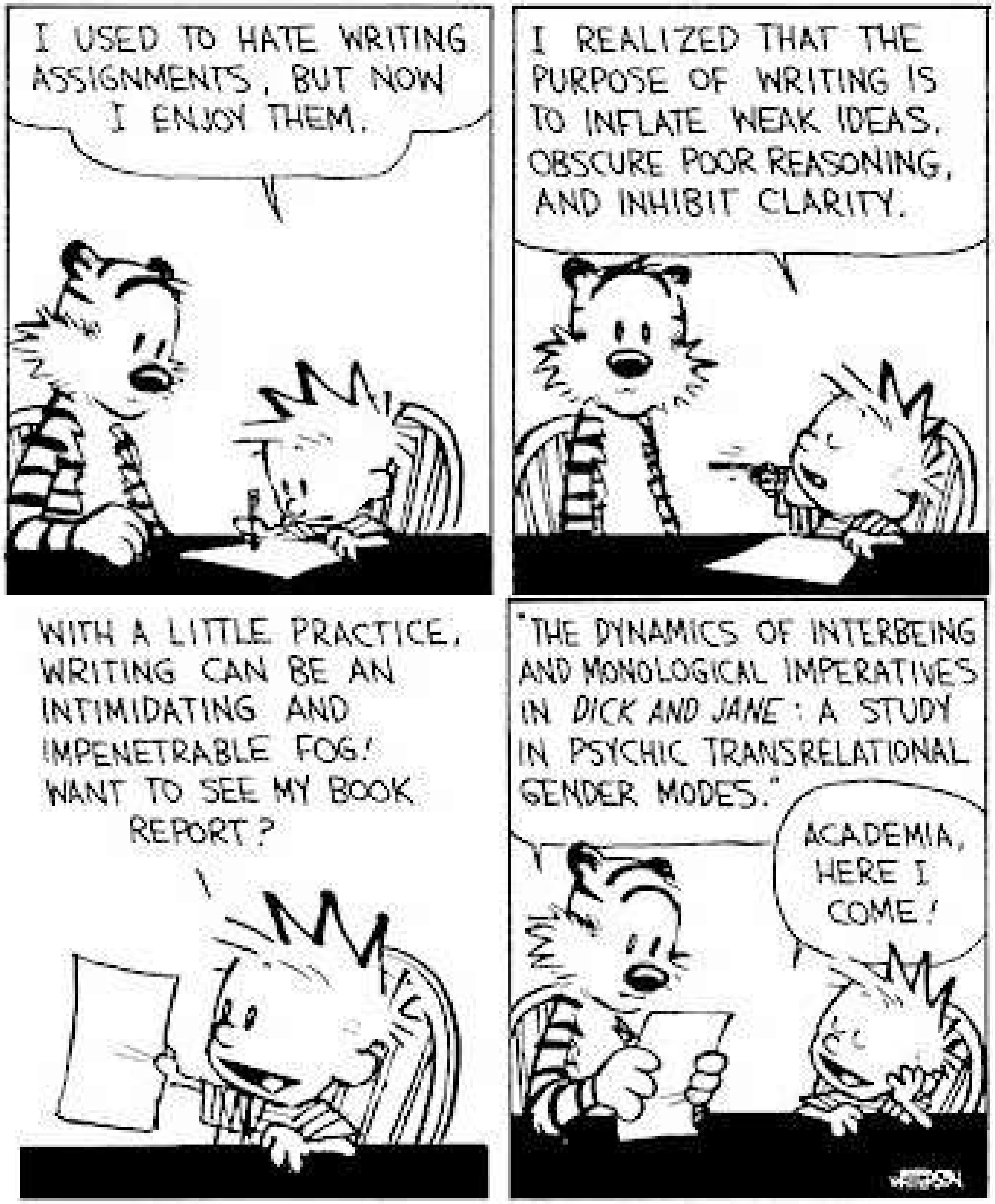
How to Write a Research Paper

Nicola Dragoni
ndra@imm.dtu.dk
Embedded Systems Engineering Group
Informatics and Mathematical Modelling
Technical University of Denmark

Slides based on a talk by Simon Peyton Jones, Microsoft Research, Cambridge

02234, DTU, Autumn 2012

Why bother?


I USED TO HATE WRITING ASSIGNMENTS, BUT NOW I ENJOY THEM.
I REALIZED THAT THE PURPOSE OF WRITING IS TO INFLATE WEAK IDEAS, OBSCURE POOR REASONING, AND INHIBIT CLARITY.
WITH A LITTLE PRACTICE, WRITING CAN BE AN INTIMIDATING AND IMPENETRABLE FOG! WANT TO SEE MY BOOK REPORT?
"THE DYNAMICS OF INTERBEING AND MONOLOGICAL IMPERATIVES IN DICK AND JANE: A STUDY IN PSYCHIC TRANSRELATIONAL GENDER MODES."
ACADEMIA, HERE I COME!
WATTERSON

1st 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

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

Do Not Be Intimidated

☑ 2nd Fallacy: you need to have a fantastic idea before you can write a paper or give a talk.

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

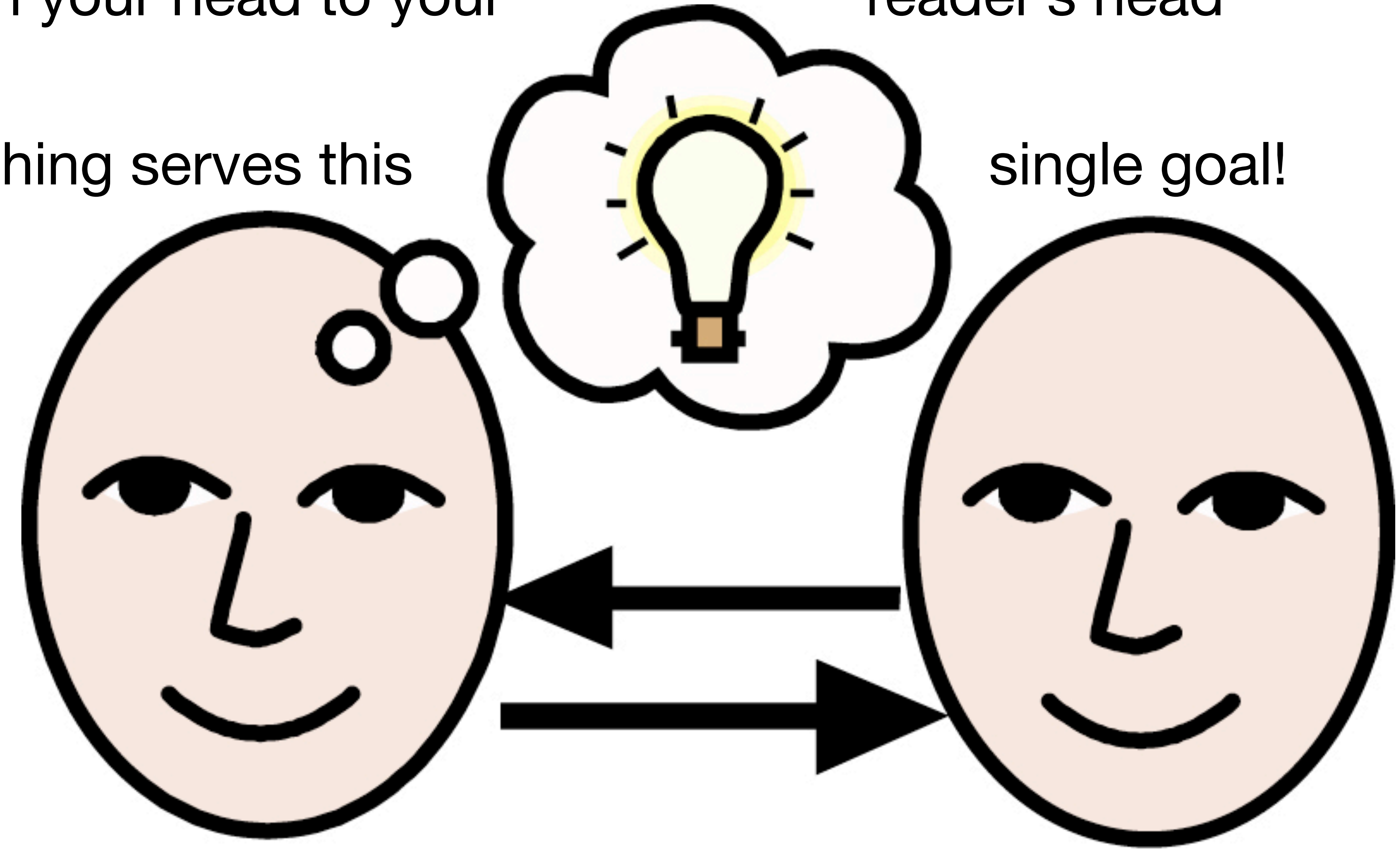
☑ Writing the paper is how you develop the idea in the first place!

The Purpose Of Your Paper

The Purpose Of Your Paper Is...

To convey your idea!

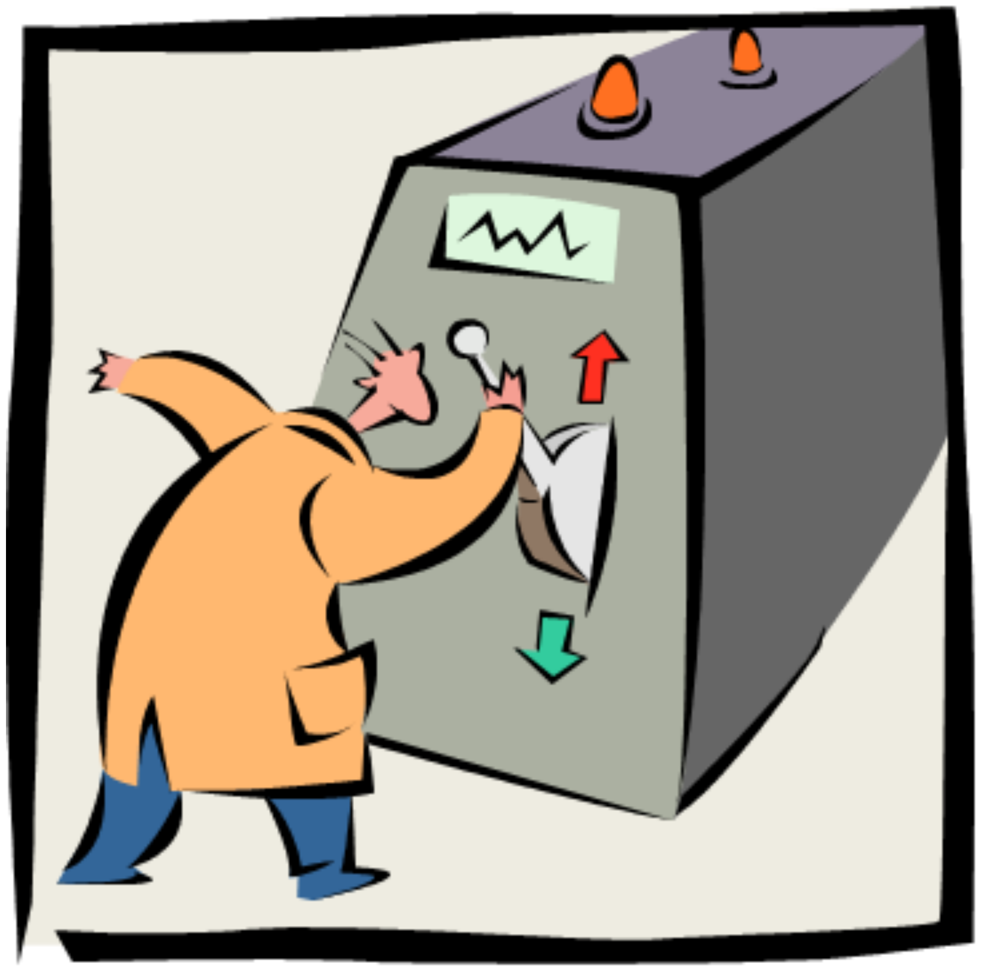
- ☑ ... from your head to your reader's head
- ☑ Everything serves this single goal!



The Purpose Of Your Paper Is NOT...

To describe the WizWoz system

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



Conveying the Idea

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

Follow Simple Guidelines...

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

Conclusion?

Related Work?

Theorems?

Contribution?

The Structure of Your Paper

Abstract?

Case study?

Introduction?

Figures?

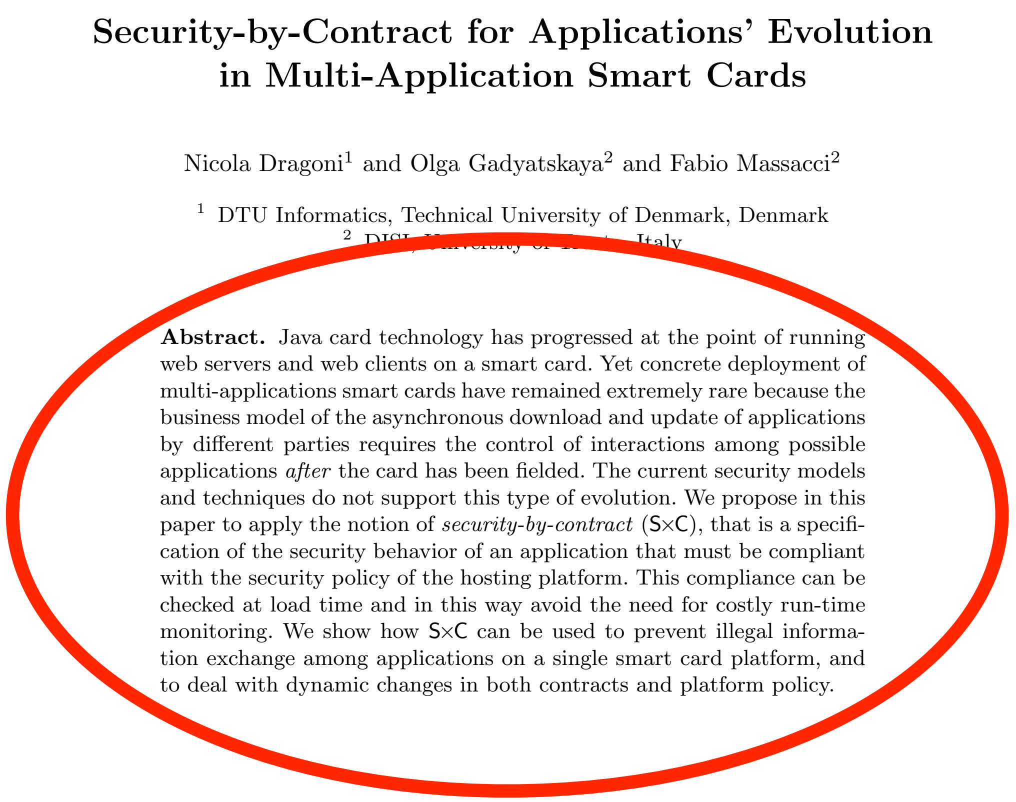
Implementation?

Structure

- ✓ Abstract (~ 5-6 sentences)
- ✓ Introduction and contribution (~ 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 Abstract

- I usually write the abstract last
- Used by program committee members to decide which papers to read
- Usually 4 “sentences”:
  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



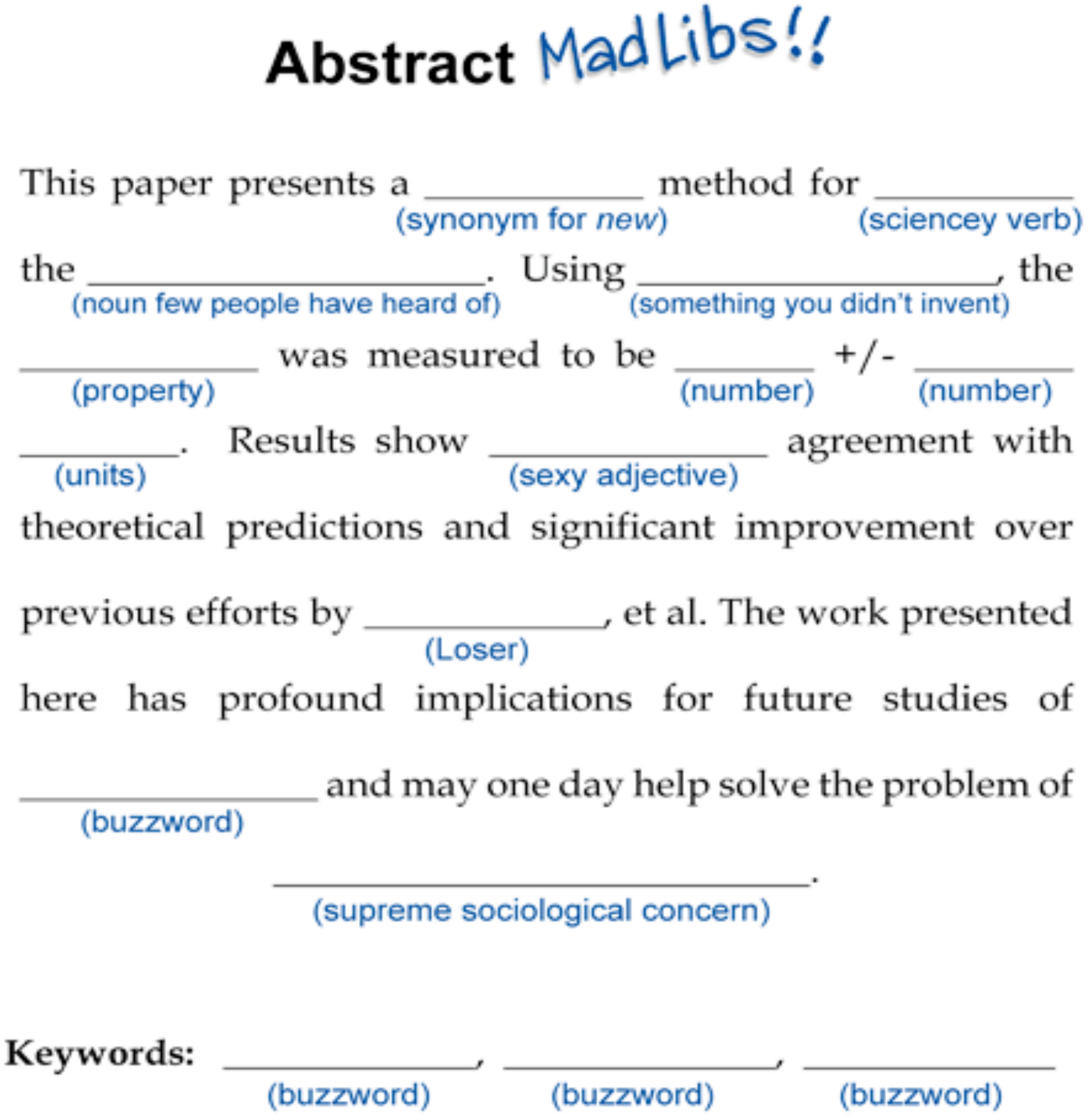
Security-by-Contract for Applications’ Evolution in Multi-Application Smart Cards

Nicola Dragoni[1] and Olga Gadyatskaya[2] and Fabio Massacci[2]

[1] DTU Informatics, Technical University of Denmark, Denmark
[2] DISI, University of Trento, Italy

Abstract. Java card technology has progressed at the point of running web servers and web clients on a smart card. Yet concrete deployment of multi-applications smart cards have remained extremely rare because the business model of the asynchronous download and update of applications by different parties requires the control of interactions among possible applications *after* the card has been fielded. The current security models and techniques do not support this type of evolution. We propose in this paper to apply the notion of *security-by-contract* (S×C), that is a specification of the security behavior of an application that must be compliant with the security policy of the hosting platform. This compliance can be checked at load time and in this way avoid the need for costly run-time monitoring. We show how S×C can be used to prevent illegal information exchange among applications on a single smart card platform, and to deal with dynamic changes in both contracts and platform policy.

Need Help?



Abstract Mad Libs!!

This paper presents a ____________ (synonym for *new*) method for ____________ (sciencey verb) the ______________________ (noun few people have heard of). Using ______________________ (something you didn't invent), the ______________ (property) was measured to be ________ (number) +/- __________ (number) __________ (units). Results show ________________ (sexy adjective) agreement with theoretical predictions and significant improvement over previous efforts by ______________ (Loser), et al. The work presented here has profound implications for future studies of __________________ (buzzword) and may one day help solve the problem of ______________________________ (supreme sociological concern).

Keywords: _______________ (buzzword), _______________ (buzzword), _______________ (buzzword)

The Introduction (1 page)

- ☑ Briefly introduce the domain of the problem
- ☑ Describe the problem (use examples!)
- ☑ Clearly and explicitly state your contributions
  - **Do not leave the reader to guess what your contributions are!**
  - Write the **list** of contributions
  - This list 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”*

Contributions Should Be Verifiable/Refutable

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.*

“Rest of this Paper is...”???

- If possible, 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.

- Someone does not like it, but I do like to have a short Outline of the Paper in the Introduction, after the Contributions:

> ***Outline of the Paper.*** *The rest of this paper is structured [organized] as follows. Section 2 introduces the problem. Section 3 describes ... Section 4 give the details of ... Section 5 presents the related work. Finally, Section 8 concludes with a summary of the main contributions of the paper”.*

Structure

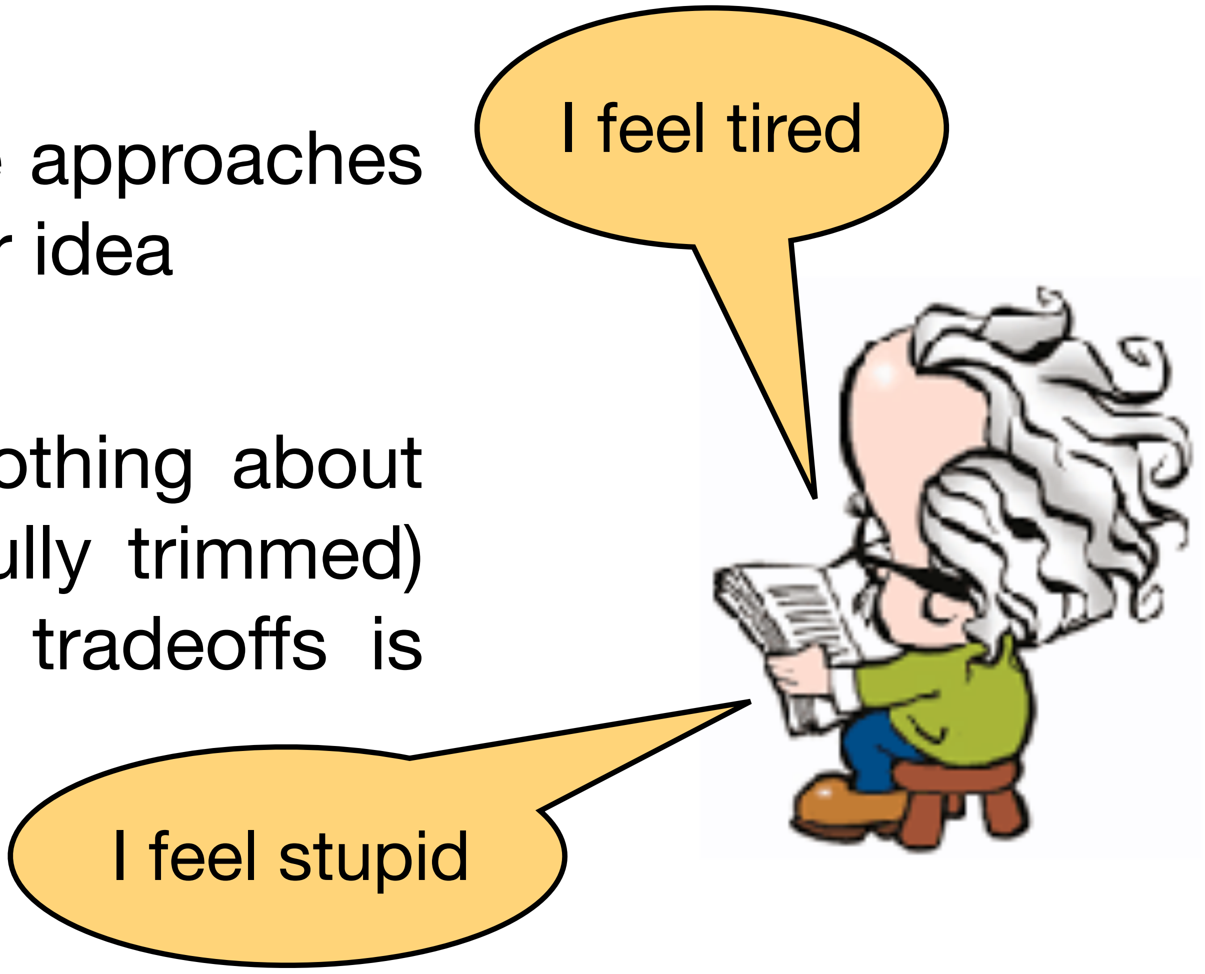
- ✓ Abstract (~ 4 sentences)
- ✓ Introduction and contribution (~ 1 page)
- ✓ The problem (~ 1 page)
- ✓ My idea (~ 2 pages)
- ✓ The details (~ 5 pages)
- ✓ Related work (~ 1-2 pages)
- ✓ Conclusions and further work (~ 0.5 pages)

Wait... Why Not 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]."

- ☑ Problem 1: describing alternative approaches gets between the reader and your idea

- ☑ Problem 2: the reader knows nothing about the problem yet; so your (carefully trimmed) description of various technical tradeoffs is absolutely <u>incomprehensible</u>



Instead...

☑ Concentrate single-mindedly on a narrative that

- Describes the problem

  why is it interesting?

- Describes your idea

- Defends your idea, showing how it solves the problem, and filling out the details

☑ On the way, cite relevant work in passing, but defer discussion to the end

Common (Big) Error: No Idea, Only Details

Consider a bufircuated 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 $p_j < p_i$.

$$D = \frac{\sum_{t=0}^{T} tCF_t DF_t}{\sum_{t=0}^{T} CF_t DF_t} = \frac{\sum_{t=0}^{T} tPV_t}{\sum_{t=0}^{T} PV_t}$$

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

Introduce the problem, and your idea, using

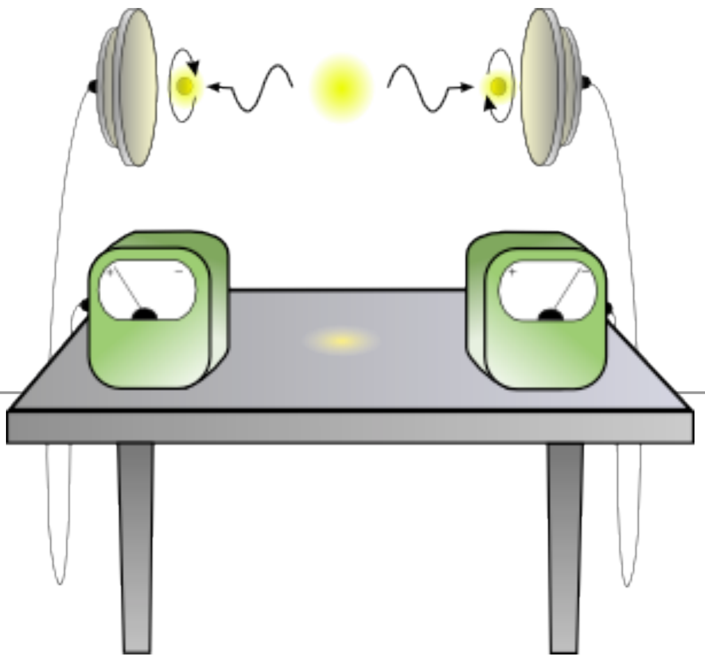
EXAMPLES

and only then present the general case!

Conveying 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, he can follow the details (but not vice versa)
- ✓ Even if he skips the details, he still takes away something valuable

But Don’t Forget Evidence!



- ☑ We are talking about **scientific** papers...
- ☑ Your introduction makes claims (list of contributions)
- ☑ The body of the paper MUST provide **evidence** to support each claim
- ☑ Evidence can be: analysis and comparison, theorems, measurements, case studies, ...
- ☑ Check each claim in the introduction, identify the evidence, and forward-reference it from the claim

$$
\begin{aligned}
\mathrm{f}(\mathbf{z}) &= \sum_{m=0}^{\infty} \frac{\mathrm{f}^{(m)}(0)}{m!}\mathbf{z}^m \\
&= \mathrm{f}(0) + \mathrm{f}'(0)\mathbf{z} + \mathrm{f}''(0)\frac{\mathbf{z}^2}{2!} + \mathrm{f}'''(0)\frac{\mathbf{z}^3}{3!} + \cdots
\end{aligned}
$$

$$
\begin{aligned}
\exp(i\alpha) &= 1 + i\alpha - \frac{\alpha^2}{2!} - i\frac{\alpha^3}{3!} + \frac{\alpha^4}{4!} + i\frac{\alpha^5}{5!} - \cdots \\
&= \left(1 - \frac{\alpha^2}{2!} + \frac{\alpha^4}{4!} - \cdots\right) + i\left(\alpha - \frac{\alpha^3}{3!} + \frac{\alpha^5}{5!} - \cdots\right) \\
&= \cos\alpha + i\sin\alpha
\end{aligned}
$$

Structure

- ✓ Abstract (~ 4 sentences)
- ✓ Introduction and contribution (~ 1 page)
- ✓ The problem (~ 1 page)
- ✓ My idea (~ 2 pages)
- ✓ The details (~ 5 pages)
- ✓ Related work (~ 1-2 pages)
- ✓ Conclusion and future work (~ 0.5 pages)

Related Work

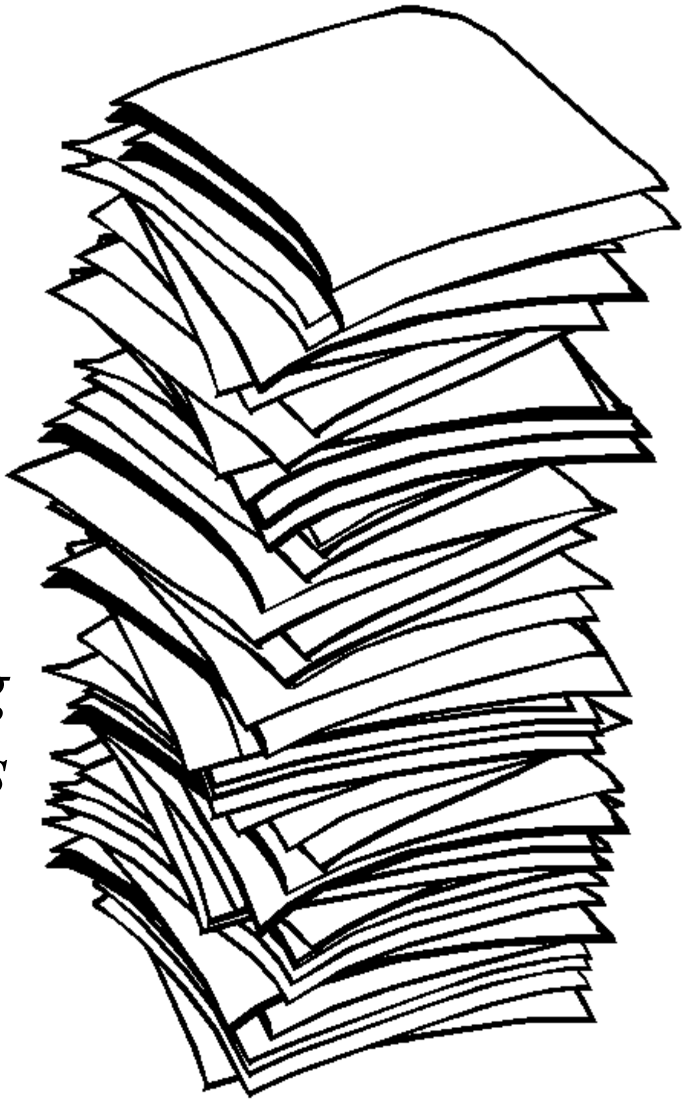
- ☑ 3rd Fallacy: to make my work look good, I have to make other people's work look bad

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

- ☑ Warmly acknowledge people who have helped you

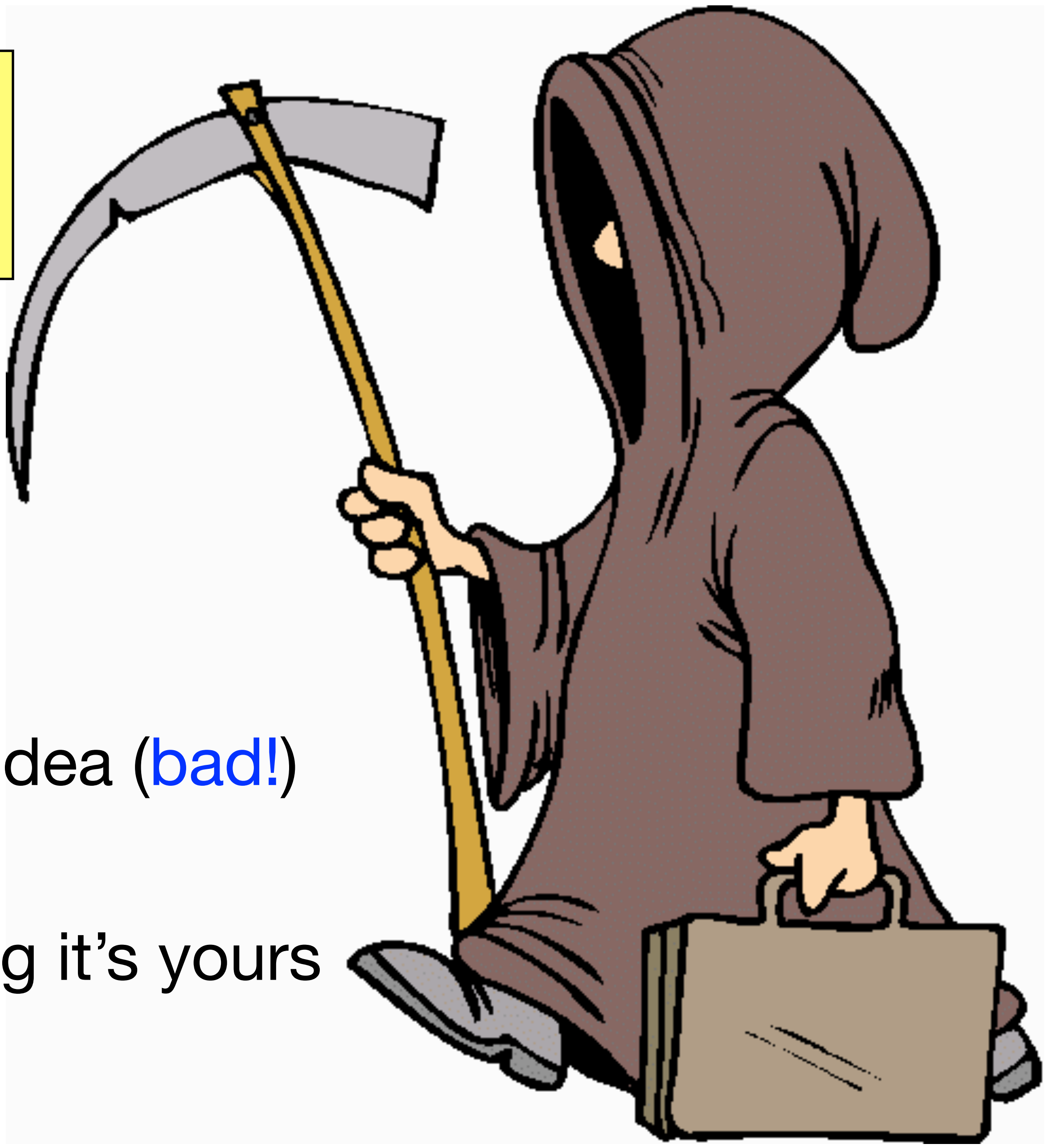
- ☑ Be generous to the competition. *"In his inspiring paper [Foo98] Foogle shows.... We develop his foundation in the following ways..."*

- ☑ Acknowledge weaknesses in your approach



Be Honest!

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

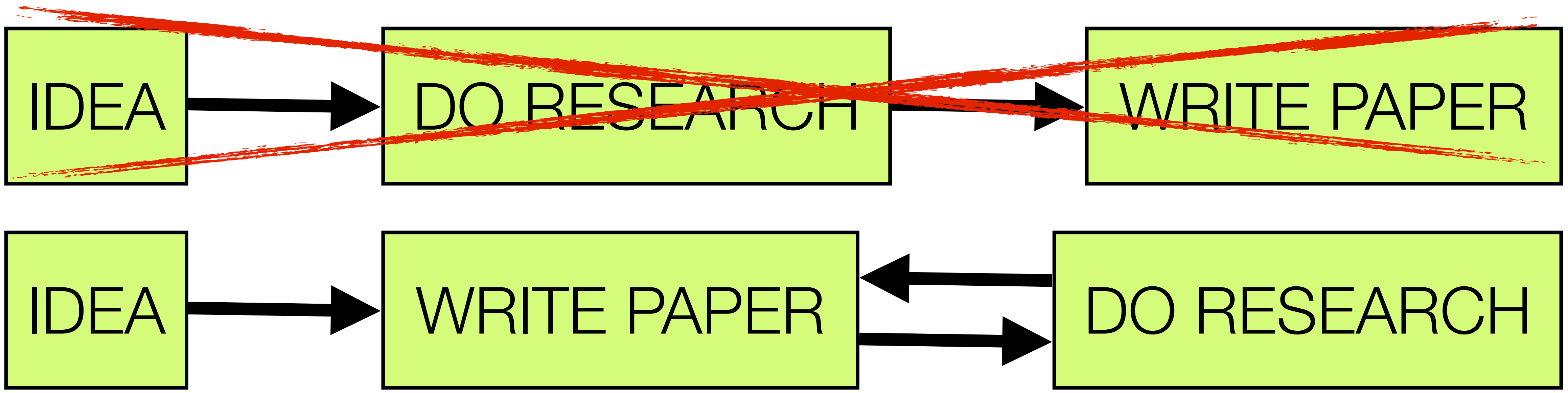
- You don't know that it's an old idea (bad!)
- You do know, but are pretending it's yours (very bad!)

Conclusion and Future Work

- ✓ Problem
- ✓ Summary of contributions
- ✓ Strengths
- ✓ Weaknesses
- ✓ Future work (possible extensions, directions to solve the weaknesses, optimizations, ...)

Other Hints

Start Early, Very Early...



- ☑ Hastily-written papers get *usually* rejected
- ☑ Papers are like wine: they need time to mature
- ☑ Collaborate
- ☑ Use CVS (or similar tools) to support collaboration

Listening to Your Reviewers

Every review is 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 important!

Listening to Your Reviewers... in Practice

- ✓ 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.

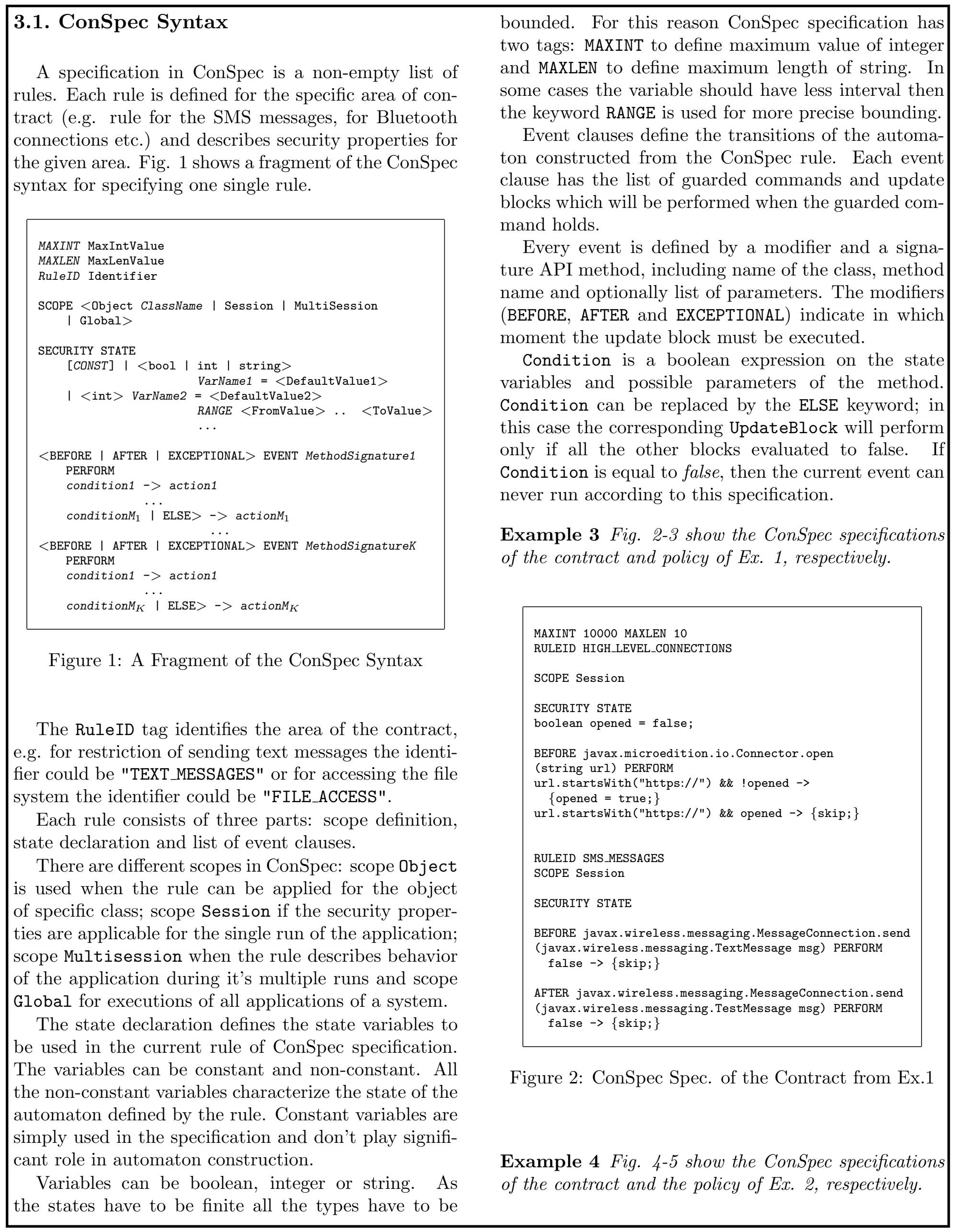
Basic (But Still Important) Stuff

- ✓ 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!



3.1. ConSpec Syntax

A specification in ConSpec is a non-empty list of rules. Each rule is defined for the specific area of contract (e.g. rule for the SMS messages, for Bluetooth connections etc.) and describes security properties for the given area. Fig. 1 shows a fragment of the ConSpec syntax for specifying one single rule.

```
MAXINT MaxIntValue
MAXLEN MaxLenValue
RuleID Identifier

SCOPE <Object ClassName | Session | MultiSession
    | Global>

SECURITY STATE
    [CONST] | <bool | int | string>
                     VarName1 = <DefaultValue1>
    | <int> VarName2 = <DefaultValue2>
                     RANGE <FromValue> ..  <ToValue>
                     ...

<BEFORE | AFTER | EXCEPTIONAL> EVENT MethodSignature1
    PERFORM
    condition1 -> action1
               ...
    conditionM1 | ELSE> -> actionM1
                       ...
<BEFORE | AFTER | EXCEPTIONAL> EVENT MethodSignatureK
    PERFORM
    condition1 -> action1
               ...
    conditionMK | ELSE> -> actionMK
```

Figure 1: A Fragment of the ConSpec Syntax

The RuleID tag identifies the area of the contract, e.g. for restriction of sending text messages the identifier could be "TEXT_MESSAGES" or for accessing the file system the identifier could be "FILE_ACCESS".

Each rule consists of three parts: scope definition, state declaration and list of event clauses.

There are different scopes in ConSpec: scope Object is used when the rule can be applied for the object of specific class; scope Session if the security properties are applicable for the single run of the application; scope Multisession when the rule describes behavior of the application during it's multiple runs and scope Global for executions of all applications of a system.

The state declaration defines the state variables to be used in the current rule of ConSpec specification. The variables can be constant and non-constant. All the non-constant variables characterize the state of the automaton defined by the rule. Constant variables are simply used in the specification and don't play significant role in automaton construction.

Variables can be boolean, integer or string. As the states have to be finite all the types have to be bounded. For this reason ConSpec specification has two tags: MAXINT to define maximum value of integer and MAXLEN to define maximum length of string. In some cases the variable should have less interval then the keyword RANGE is used for more precise bounding.

Event clauses define the transitions of the automaton constructed from the ConSpec rule. Each event clause has the list of guarded commands and update blocks which will be performed when the guarded command holds.

Every event is defined by a modifier and a signature API method, including name of the class, method name and optionally list of parameters. The modifiers (BEFORE, AFTER and EXCEPTIONAL) indicate in which moment the update block must be executed.

Condition is a boolean expression on the state variables and possible parameters of the method. Condition can be replaced by the ELSE keyword; in this case the corresponding UpdateBlock will perform only if all the other blocks evaluated to false. If Condition is equal to *false*, then the current event can never run according to this specification.

Example 3 *Fig. 2-3 show the ConSpec specifications of the contract and policy of Ex. 1, respectively.*

```
MAXINT 10000 MAXLEN 10
RULEID HIGH_LEVEL_CONNECTIONS

SCOPE Session

SECURITY STATE
boolean opened = false;

BEFORE javax.microedition.io.Connector.open
(string url) PERFORM
url.startsWith("https://") && !opened ->
  {opened = true;}
url.startsWith("https://") && opened -> {skip;}


RULEID SMS_MESSAGES
SCOPE Session

SECURITY STATE

BEFORE javax.wireless.messaging.MessageConnection.send
(javax.wireless.messaging.TextMessage msg) PERFORM
  false -> {skip;}

AFTER javax.wireless.messaging.MessageConnection.send
(javax.wireless.messaging.TestMessage msg) PERFORM
  false -> {skip;}
```

Figure 2: ConSpec Spec. of the Contract from Ex.1

Example 4 *Fig. 4-5 show the ConSpec specifications of the contract and the policy of Ex. 2, respectively.*

Example

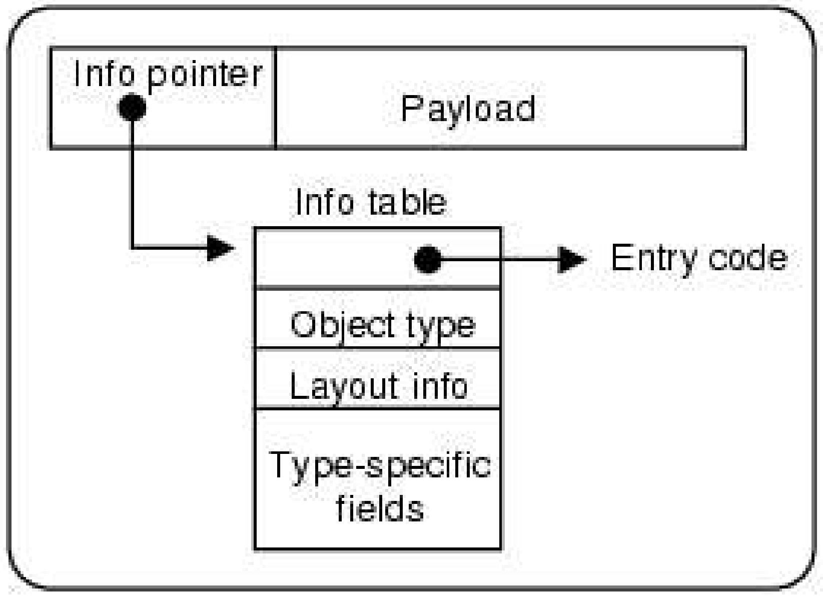


Figure 3. A heap object

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 if possible.

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

Summary

- ✓ If you remember nothing else:
  - ✓ Identify your key idea
  - ✓ Make your contributions explicit
  - ✓ Use examples