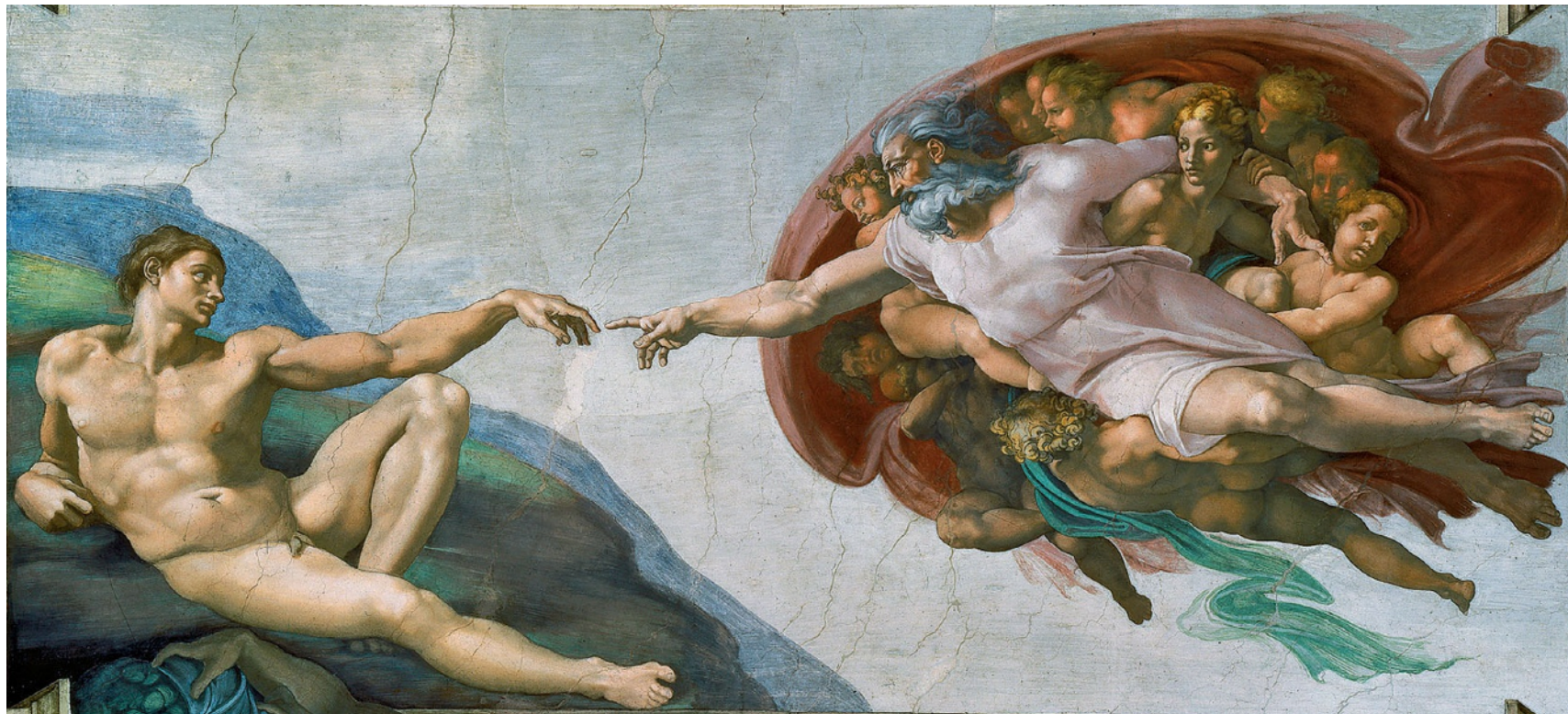


# HOW TO WRITE PAPERS SO PEOPLE CAN READ THEM



**Derek Dreyer**

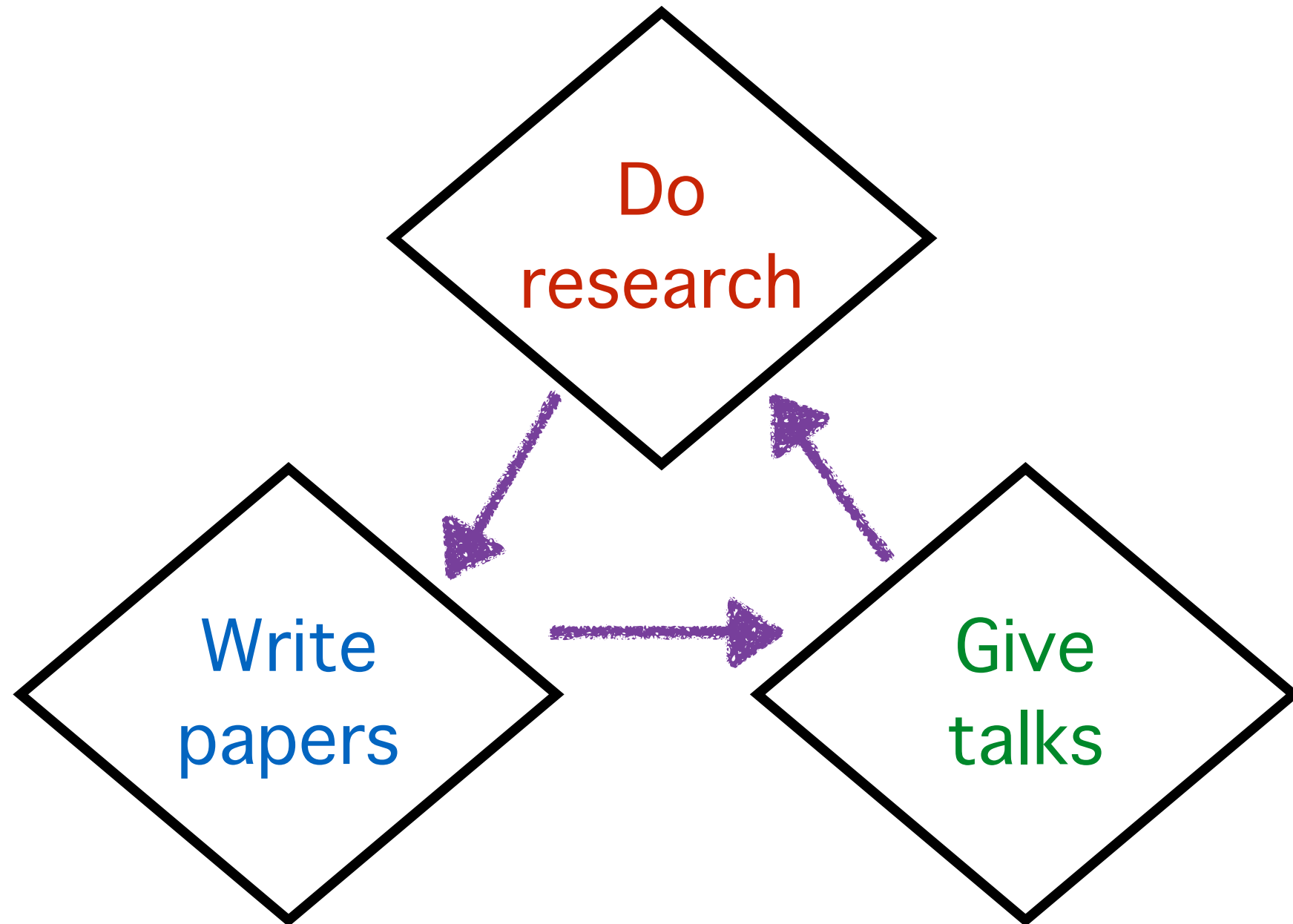
Max Planck Institute for Software Systems

*PLMW@ICFP 2020  
Not Jersey City, USA*

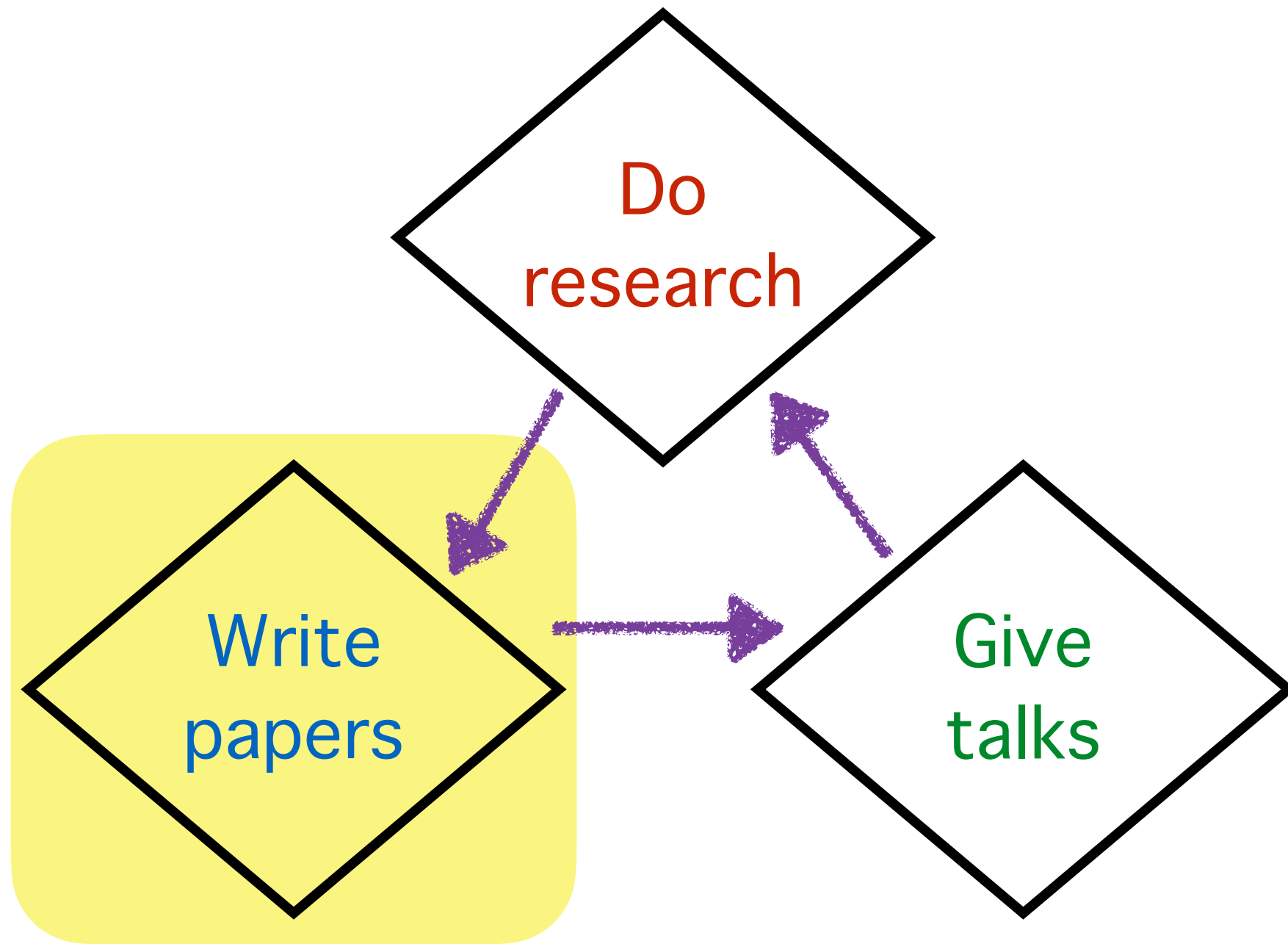
# My job as a researcher



# My job as a researcher



# My job as a researcher





Have you read any  
research papers lately?

# Have you read any research papers lately?



# Have you read any research papers lately?

- You may think you just lack the technical sophistication to understand them.



# Have you read any research papers lately?

- You may think you just lack the technical sophistication to understand them.
- But in fact, many papers are **poorly written**.



# So if you can write clear, accessible papers...

- People will **enjoy** reading them!
- People will **learn** something from them!
- They will get **accepted** to top conferences!

*Fame*





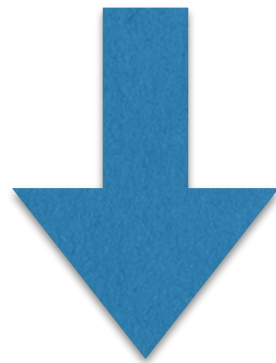
# So if you can write clear, accessible papers...

- People will **enjoy** reading them!
- People will **learn** something from them!
- They will get **accepted** to top conferences!

*Fame*



A piece of research



Writer



Reader

By downcasting the pre-axial gaskets,  
we achieved 47% reduction in XPS latency  
on the re-uptake bivalve!



Writer



Reader

By downcasting the pre-axial gaskets,  
we achieved 47% reduction in XPS latency  
on the re-uptake bivalve!

OK, but what does it **do**,  
and why do I **care**?



Writer



Reader



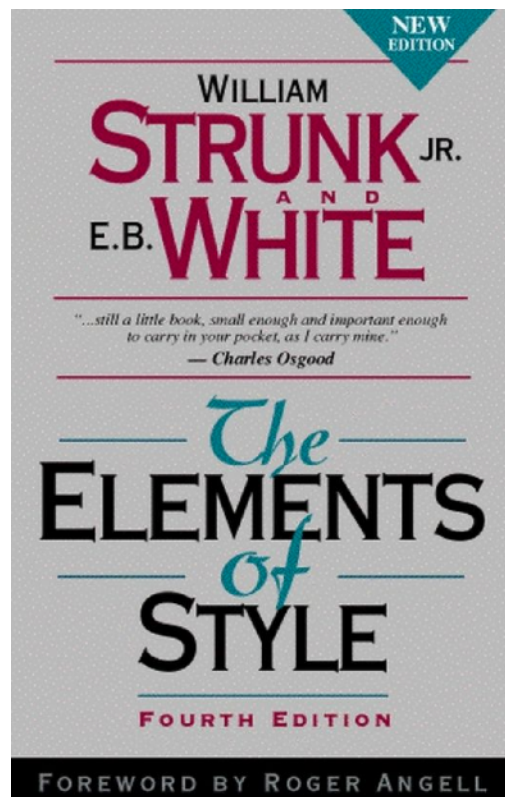
# The good news

- There are **principles** you can follow that will help you write clearer, more readable prose
  - Based on how readers process information



# The good news

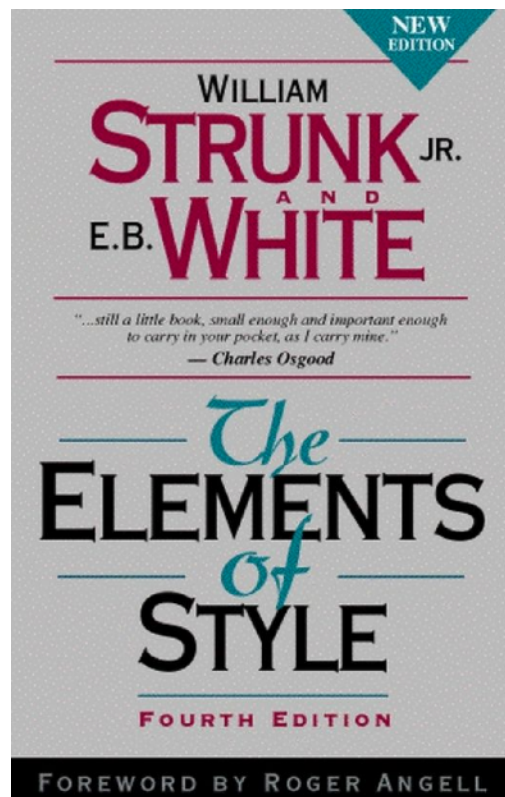
- There are **principles** you can follow that will help you write clearer, more readable prose
  - Based on how readers process information



?

# The good news

- There are **principles** you can follow that will help you write clearer, more readable prose
  - Based on how readers process information



?

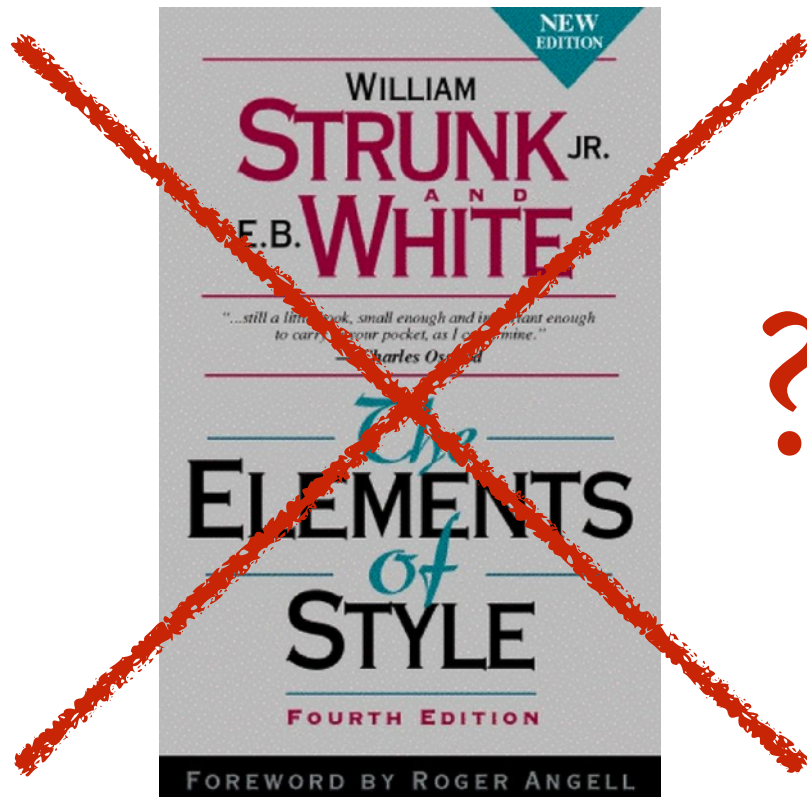
“Be clear”

“Omit needless words”

...

# The good news

- There are **principles** you can follow that will help you write clearer, more readable prose
  - Based on how readers process information



?

“Be clear”

“Omit needless words”

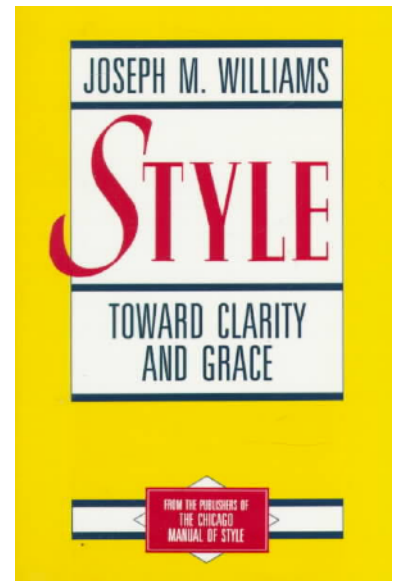
...

# The good news

- There are **principles** you can follow that will help you write clearer, more readable prose
  - Based on how readers process information
- These principles are **constructive**:
  - Easy to check if your text satisfies these principles
  - If not, principles suggest improvements

# Inspirations for this talk

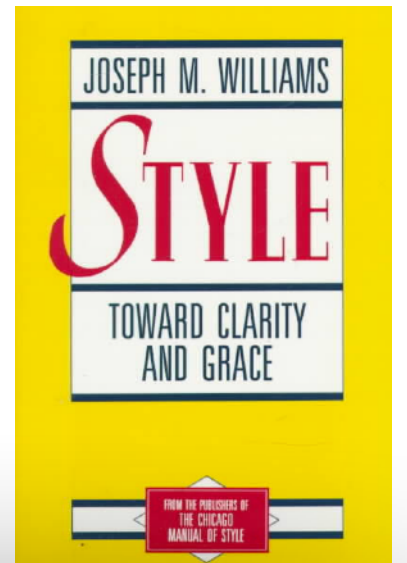
- **Joseph M. Williams.** *Style: Toward clarity and grace.* 1990. (book)
- **Norman Ramsey.** *Learn technical writing in two hours per week.* (course notes)
  - <http://www.cs.tufts.edu/~nr/pubs/two.pdf>
- **Simon Peyton Jones.** *How to write a great research paper.* (talk)
  - <https://www.microsoft.com/en-us/research/video/how-to-write-a-great-research-paper-3/>





# Inspirations for this talk

- **Joseph M. Williams.** *Style: Toward clarity and grace.* 1990. (book)



Talk developed jointly with  
**Rose Hoberman**  
@ MPI-SWS



- **Simon Peyton Jones.** *How to write a great research paper.* (talk)

- <https://www.microsoft.com/en-us/research/video/how-to-write-a-great-research-paper-3/>



Sentences & paragraphs

# Flow



It should be clear how each sentence and paragraph relates to **the adjacent ones**

Does this text flow?

# Does this text flow?

Security proofs of cryptographic protocols are crucial for the security of everyday electronic communication. However, these proofs tend to be complex and difficult to get right. The game-playing technique, originally proposed by Jones et al., follows a code-based approach where the security properties are formulated in terms of probabilistic programs, called games. This is a general design principle for cryptographic proofs to ease their management.



# Does this text flow?

Security proofs of cryptographic protocols are crucial for the security of everyday electronic communication.

However, these proofs tend to be complex and difficult to get right. The game-playing technique, originally proposed by Jones et al., follows a code-based approach where the security properties are formulated in terms of probabilistic programs, called games. This is a general design principle for cryptographic proofs to ease their management.

# Does this text flow?

Security proofs of cryptographic protocols are crucial for the security of everyday electronic communication.

However, these proofs tend to be complex and difficult to get right.

The game-playing technique, originally proposed by Jones et al., follows a code-based approach where the security properties are formulated in terms of probabilistic programs, called games. This is a general design principle for cryptographic proofs to ease their management.

# Does this text flow?

Security proofs of cryptographic protocols are crucial for the security of everyday electronic communication. However, these proofs tend to be complex and difficult to get right. The game-playing technique, originally proposed by Jones et al., follows a code-based approach where the security properties are formulated in terms of probabilistic programs, called games. This is a general design principle for cryptographic proofs to ease their management.

# Does this text flow?

Security proofs of cryptographic protocols are crucial for the security of everyday electronic communication. However, these proofs tend to be complex and difficult to get right. The game-playing technique, originally proposed by Jones et al., follows a code-based approach where the security properties are formulated in terms of probabilistic programs, called games. This is a general design principle for cryptographic proofs to ease their management.

# Does this text flow?

Security proofs of cryptographic protocols are crucial for the security of everyday electronic communication. However, these proofs tend to be complex and difficult to get right. The game-playing technique, originally proposed by Jones et al., follows a code-based approach where the security properties are formulated in terms of probabilistic programs, called games. This is a general design principle for cryptographic proofs to ease their management.

# Does this text flow?

Security proofs of cryptographic protocols are crucial for the security of everyday electronic communication. However, these proofs tend to be complex and difficult to get right. The game-playing technique, originally proposed by Jones et al., follows a code-based approach where the security properties are formulated in terms of probabilistic programs, called games. This is a general design principle for cryptographic proofs to ease their management.

# Does this text flow?

Security proofs of cryptographic protocols are crucial for the security of everyday electronic communication.

However, these proofs tend to be complex and difficult to get right.

The game-playing technique, originally proposed by Jones et al., follows a code-based approach



**What does this game-playing technique have to do with what came before?**



# Old to new

- Begin sentences with old info
  - Creates link to earlier text
- End sentences with new info
  - Creates link to the text that follows
  - Also places new info in position of **emphasis**



# Applying old-to-new

## New information

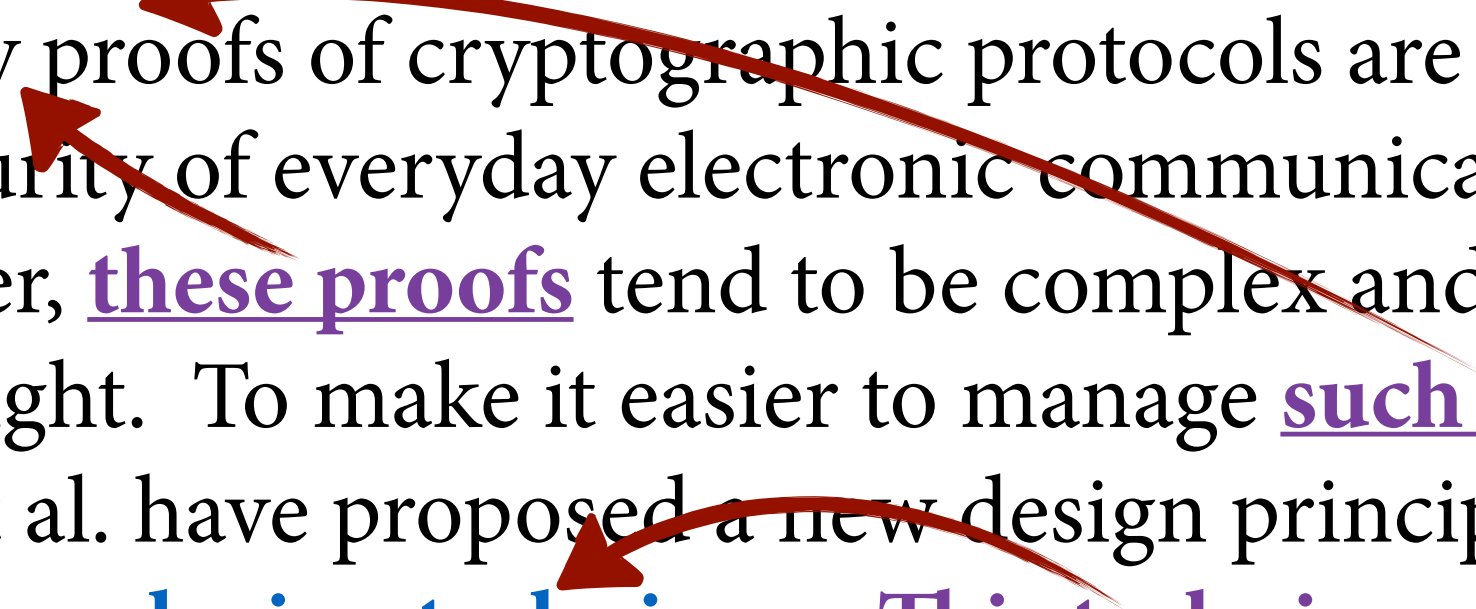
Security proofs of cryptographic protocols are crucial for the security of everyday electronic communication. However, these proofs tend to be complex and difficult to get right. **The game-playing technique, originally proposed by Jones et al., follows a code-based approach where the security properties are formulated in terms of probabilistic programs, called games.** This is a general design principle for cryptographic proofs to ease their management.

# Applying old-to-new

Security proofs of cryptographic protocols are crucial for the security of everyday electronic communication. However, these proofs tend to be complex and difficult to get right. To make it easier to manage such proofs, Jones et al. have proposed a new design principle, called the game-playing technique. This technique follows a code-based approach where the security properties are formulated in terms of probabilistic programs, called games.

# Old-to-new satisfied

Security proofs of cryptographic protocols are crucial for the security of everyday electronic communication. However, these proofs tend to be complex and difficult to get right. To make it easier to manage such proofs, Jones et al. have proposed a new design principle, called the **game-playing technique**. This technique follows a code-based approach where the security properties are formulated in terms of probabilistic programs, called games.



The diagram consists of two red curved arrows. The first arrow starts from the underlined phrase 'these proofs' and points to the underlined phrase 'such proofs'. The second arrow starts from the underlined phrase 'This technique' and points back to the underlined phrase 'these proofs'. This visualizes a cycle of satisfaction where the new technique satisfies the requirements of the old proofs.

But flow is not enough!

# What about this text?

Lions and tigers are some of the most dramatic and awe-inspiring species of cats. Most of these large cats, however, are currently facing extinction. A smaller cat that has been more evolutionarily successful is the house cat. Although house cats are currently the most popular pet in the world, they are in many ways anti-social. It would therefore be interesting to study whether house cats can be trained to be more sociable.

# What about this text?

Lions and tigers are some of the most dramatic and awe-inspiring species of cats. Most of these large cats, however, are currently facing extinction. A smaller cat that has been more evolutionarily successful is the house cat. Although house cats are currently the most popular pet in the world, they are in many ways anti-social. It would therefore be interesting to study whether house cats can be trained to be more sociable.



# What about this text?

Lions and tigers are some of the most dramatic and awe-inspiring species of cats. **Most of these large cats, however, are currently facing extinction.** A smaller cat that has been more evolutionarily successful is the house cat. Although house cats are currently the most popular pet in the world, they are in many ways anti-social. It would therefore be interesting to study whether house cats can be trained to be more sociable.

# What about this text?

Lions and tigers are some of the most dramatic and awe-inspiring species of cats. Most of these large cats, however, are currently facing extinction. **A smaller cat that has been more evolutionarily successful is the house cat.** Although house cats are currently the most popular pet in the world, they are in many ways anti-social. It would therefore be interesting to study whether house cats can be trained to be more sociable.

# What about this text?

Lions and tigers are some of the most dramatic and awe-inspiring species of cats. Most of these large cats, however, are currently facing extinction. A smaller cat that has been more evolutionarily successful is the house cat. **Although house cats are currently the most popular pet in the world, they are in many ways anti-social.** It would therefore be interesting to study whether house cats can be trained to be more sociable.

# What about this text?

Lions and tigers are some of the most dramatic and awe-inspiring species of cats. Most of these large cats, however, are currently facing extinction. A smaller cat that has been more evolutionarily successful is the house cat. Although house cats are currently the most popular pet in the world, they are in many ways anti-social. It would therefore be interesting to study whether house cats can be trained to be more sociable.

# What about this text?

Lions and tigers are some of the most dramatic and awe-inspiring species of cats. Most of these large cats, however, are currently facing extinction. A smaller cat that has been more evolutionarily successful is the house cat. Although house cats are currently the most popular pet in the world, they are in many ways anti-social. It would therefore be interesting to study whether house cats can be trained to be more sociable.

# What about this text?

Lions and tigers are some of the most dramatic and awe-inspiring species of cat. The large cats, however, are not the only ones. A smaller, less dramatic, but equally successful species is the house cat. House cats are currently the most popular pet in the world, and they are increasingly becoming a social animal. It would therefore be interesting to study whether house cats can be trained to be more sociable.

**Has great flow,  
but is incoherent!**



# Coherence



It should be clear how each  
sentence and paragraph relates to  
**the big picture**



# One paragraph, one point

- A paragraph should have one main point, expressed in a single **point sentence**
- **Typically** the point sentence should appear **at or near the beginning of the paragraph**



Get to the  
**point!**

# No point sentence

Lions and tigers are some of the most dramatic and awe-inspiring species of cats. Most of these large cats, however, are currently facing extinction. A smaller cat that has been more evolutionarily successful is the house cat. Although house cats are currently the most popular pet in the world, they are in many ways anti-social. It would therefore be interesting to study whether house cats can be trained to be more sociable.

# Point sentence up front

There appears to be a negative correlation between the charisma of a species and its ability to survive.

Lions and tigers, for instance, are among the most majestic creatures in the animal kingdom, yet they are currently facing extinction. In contrast, the house cat is evolutionarily quite successful, even though it is mostly known for stupid pet tricks.

# Flow & coherence



Create **flow** with **old to new**

Create **coherence** with  
**one paragraph, one point**



# Two other principles



- **Name your baby:**
  - Give unique names to things and use them consistently



- **Just in time:**
  - Give information precisely when it is needed, not before



# Three other principles



Bonus principle from Rose

**Short subjects:**

Subject of sentence should be  
at most 8 words long

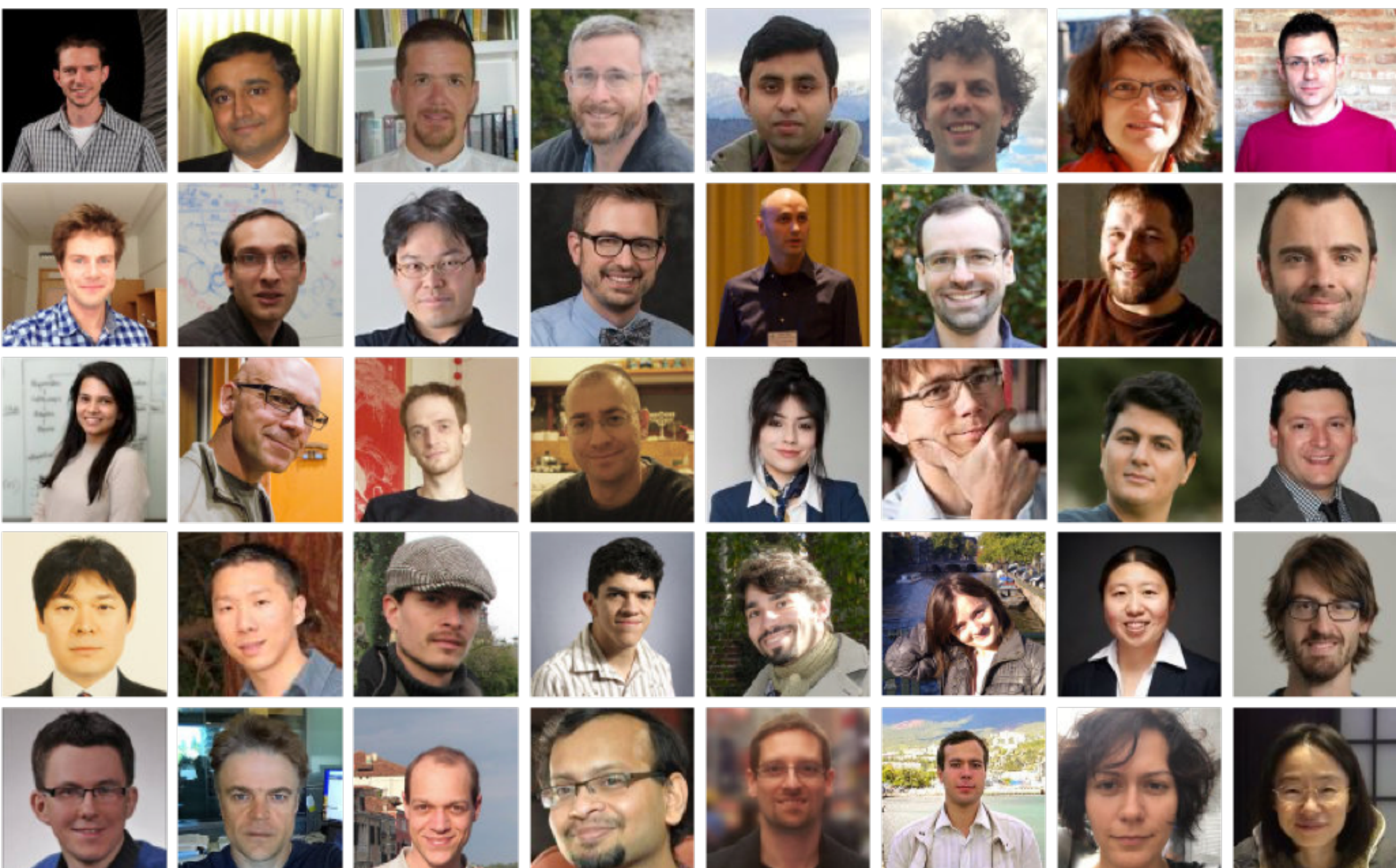


- Give information precisely  
when it is needed, not before



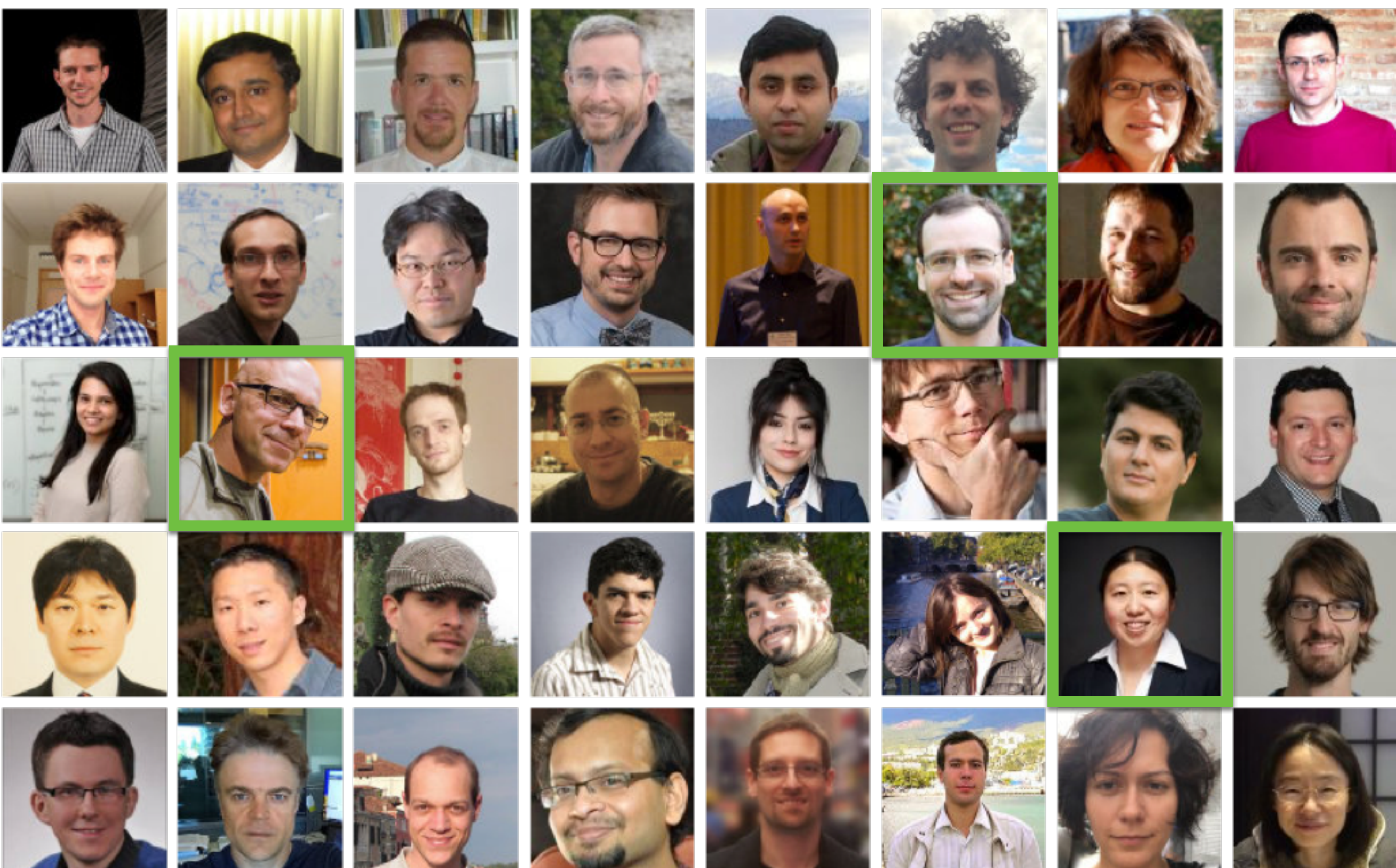
# Structure of a research paper

# A Typical Program Committee





# A Typical Program Committee





# A Typical Program Committee

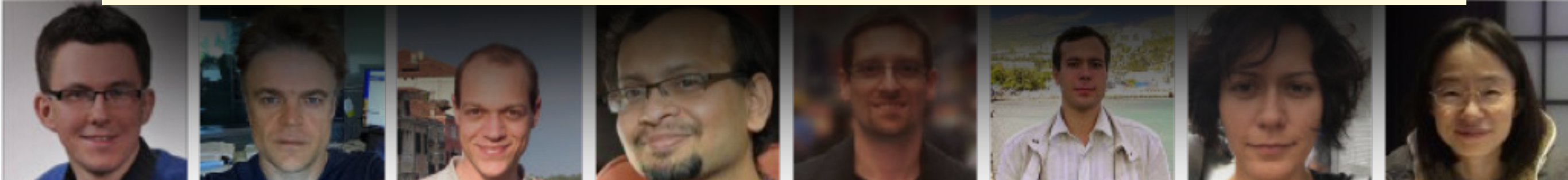
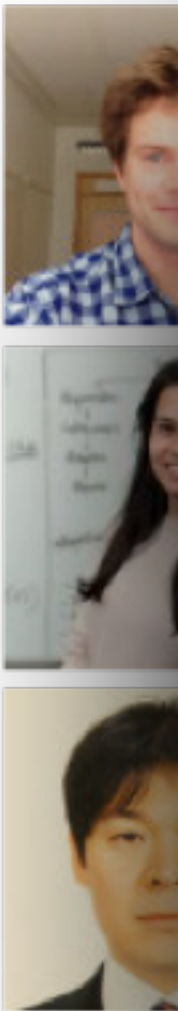


**PC is diverse:**

- Your reviewers may not be “experts”

**Review period is short (~2 months):**

- ~15-20 papers per PC member
- Often < 1 day to review each paper



*Overarching Principle #1*

# TOP-DOWN

Explain your work at multiple levels of abstraction,  
starting at a high level (accessible to non-experts)  
and getting progressively more detailed

*Overarching Principle #2*

Tell them what they  
want to know



## Overarching Principle #2

Tell them what they  
want to know



- ☑ How is your work **important**?
- ☑ How is your work **novel**?
- ☑ How is your work **interesting**?



## Overarching Principle #2

Tell them what they  
want to know



- ☒ How is your work **important**?
- ☒ How is your work **novel**?
- ☒ How is your work **interesting**?
- ☐ ~~How was your work **challenging**?~~

# A structure that works

- **Abstract** (1-2 paragraphs, 1000 readers)
- **Intro** (2-4 pages, 100 readers)
- **Key ideas** (4-6 pages, 50 readers)
- **Technical meat** (8-12 pages, 5 readers)
- **Related work** (1-3 pages, 100 readers)

# A structure that works

- **Abstract** (1-2 paragraphs, 1000 readers)
- **Intro** (2-4 pages, 100 readers)
- Key ideas (4-6 pages, 50 readers)
- Technical meat (8-12 pages, 5 readers)
- Related work (1-3 pages, 100 readers)

# The CGI model for an abstract/intro

- **Context:**

- Set the stage, motivate the general topic


- **Gap:**

- Explain your specific problem and why existing work does not adequately solve it

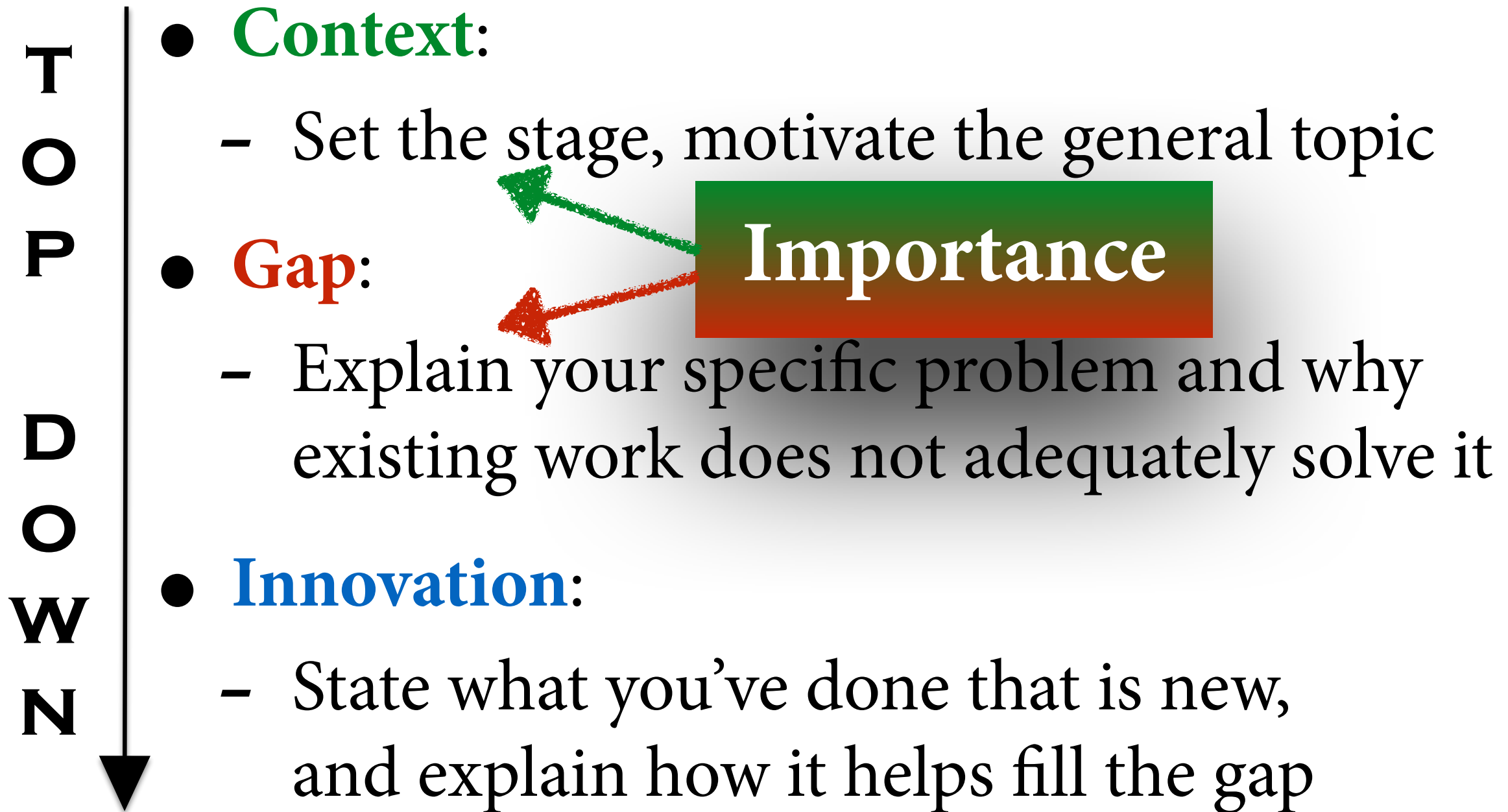
- **Innovation:**

- State what you've done that is new, and explain how it helps fill the gap

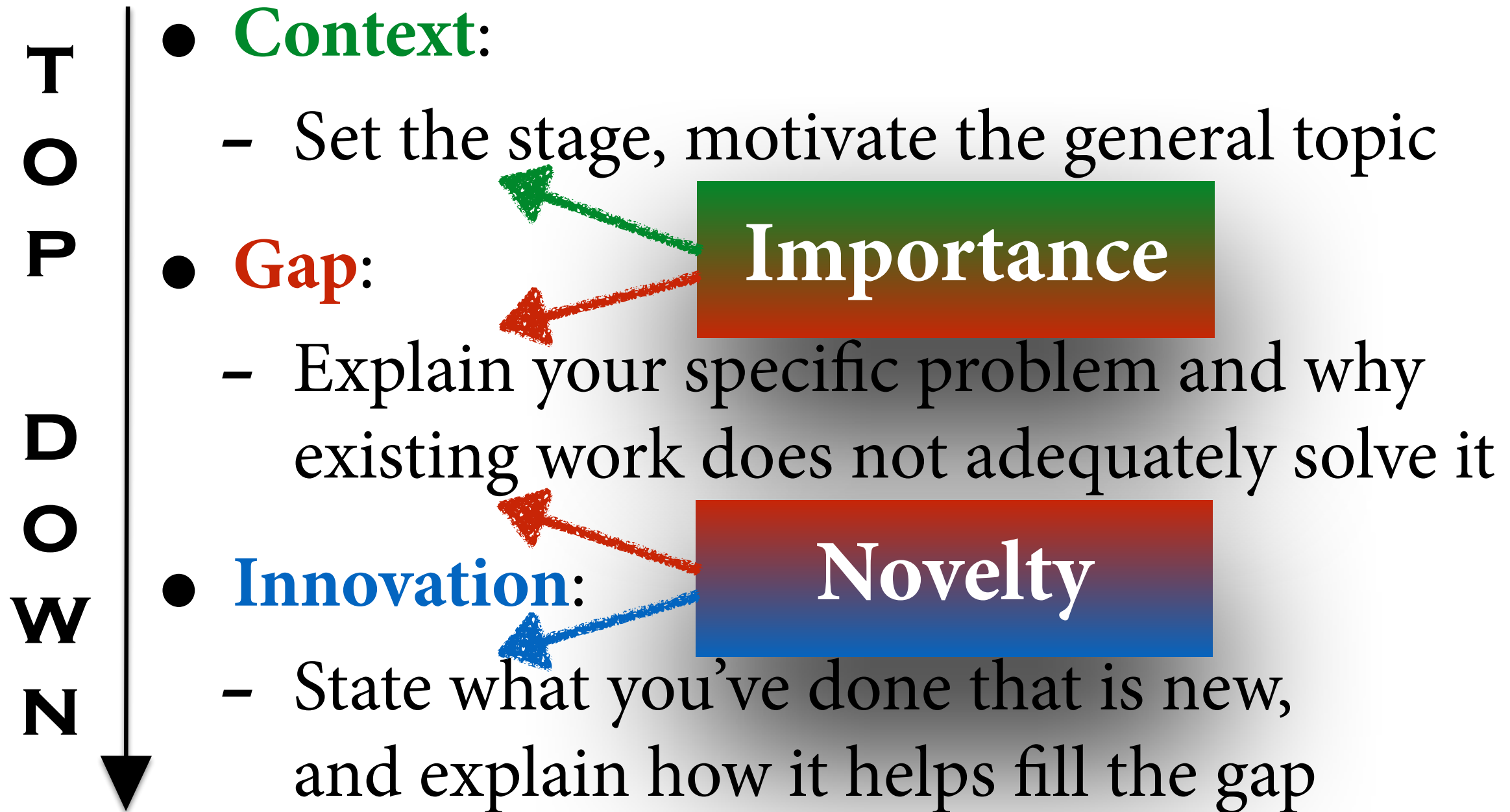
# The CGI model for an abstract/intro

- T  
O  
P**  
**D  
O  
W  
N**
- 
- **Context:**
    - Set the stage, motivate the general topic
  - **Gap:**
    - Explain your specific problem and why existing work does not adequately solve it
  - **Innovation:**
    - State what you've done that is new, and explain how it helps fill the gap

# The CGI model for an abstract/intro



# The CGI model for an abstract/intro





*An abstract for this talk*

# Context

Learning to write well is an essential part of becoming a successful researcher.

# Gap

Learning to write well is an essential part of becoming a successful researcher. Unfortunately, many researchers find it very hard to write well because they do not know how to view their text from the perspective of the reader.

# Innovation

Learning to write well is an essential part of becoming a successful researcher. Unfortunately, many researchers find it very hard to write well because they do not know how to view their text from the perspective of the reader. In this talk, we present a simple set of principles for good writing, based on an understanding of how readers process information. Unlike such platitudes as "Be clear" or "Omit needless words", our principles are *constructive*: one can easily check whether a piece of text satisfies them, and if it does not, the principles suggest concrete ways to improve it.

# Introduction

- Like an expanded version of the abstract
- Alternative approach (SPJ): Eliminate **C**ontext
  - Start with a concrete example, e.g. “Consider this Haskell code...”
  - If this works, it can be effective, but I find it often doesn’t work
  - It assumes reader already knows context



# A structure that works

- Abstract (1-2 paragraphs, 1000 readers)
- Intro (2-4 pages, 100 readers)
- **Key ideas** (4-6 pages, 50 readers)
- Technical meat (8-12 pages, 5 readers)
- Related work (1-3 pages, 100 readers)



# Why have a “key ideas” section at all?



1. Forces you to have a **takeaway**, i.e. something **interesting**!
2. Many readers only care about the takeaway, not the technical details
3. For those who want the technical details, the key ideas are still useful as “scaffolding”

# “Key ideas” section



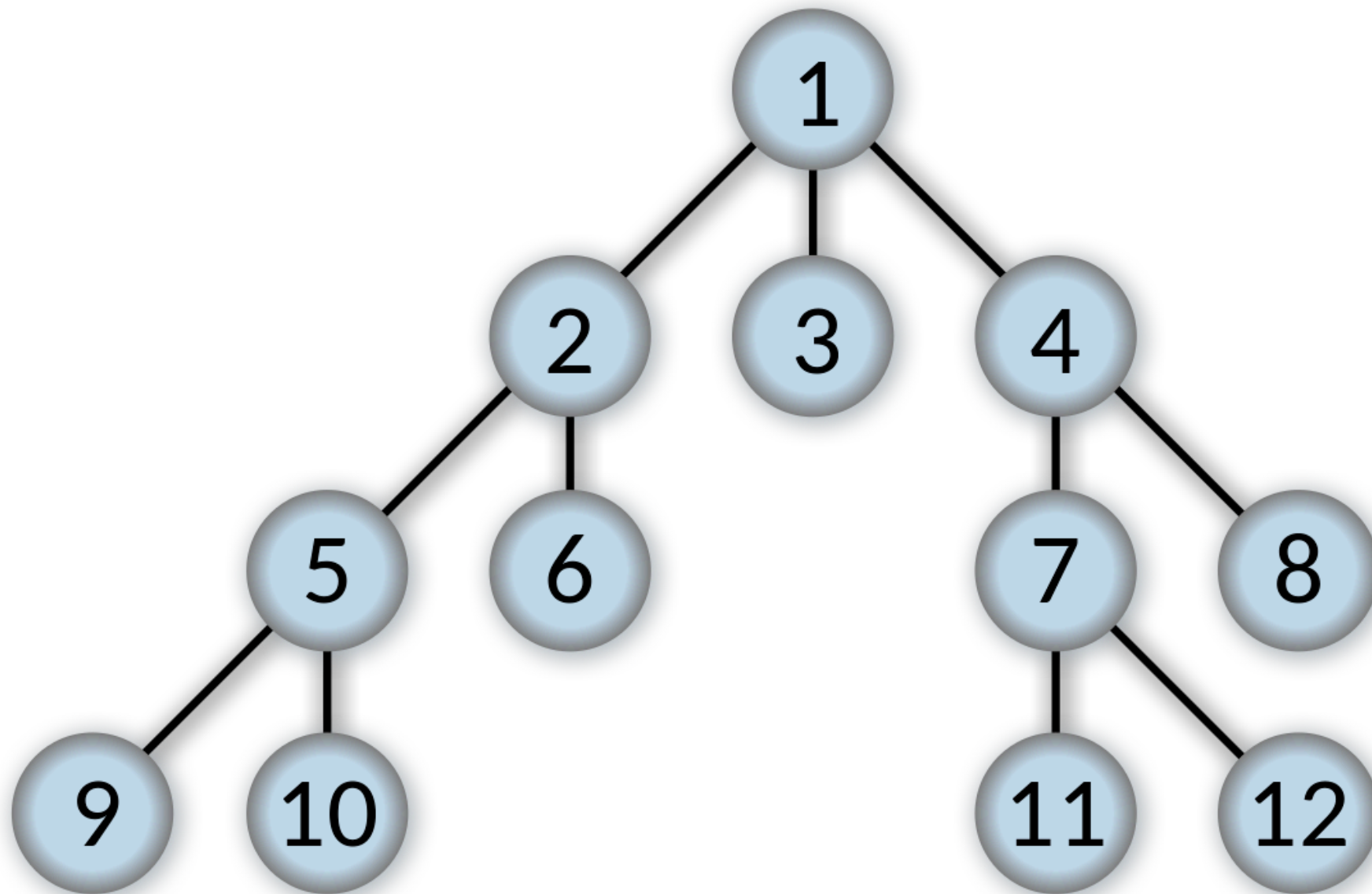
- Use **concrete illustrative examples** and high-level intuition
- Do **not** have to show the general solution (that's what the technical section is for)

# A confession

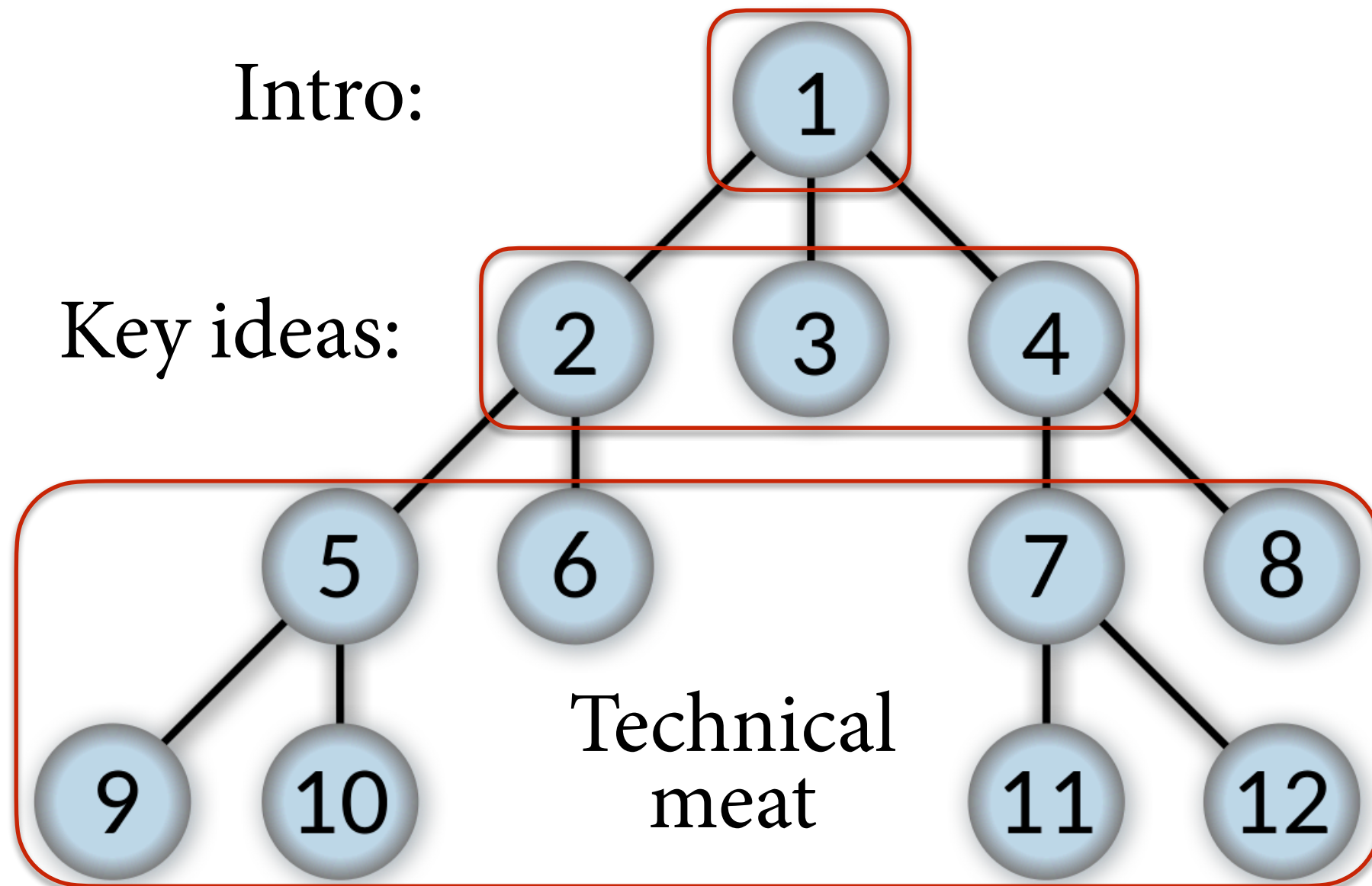


I don't always have a key ideas section.

# Breadth-first traversal



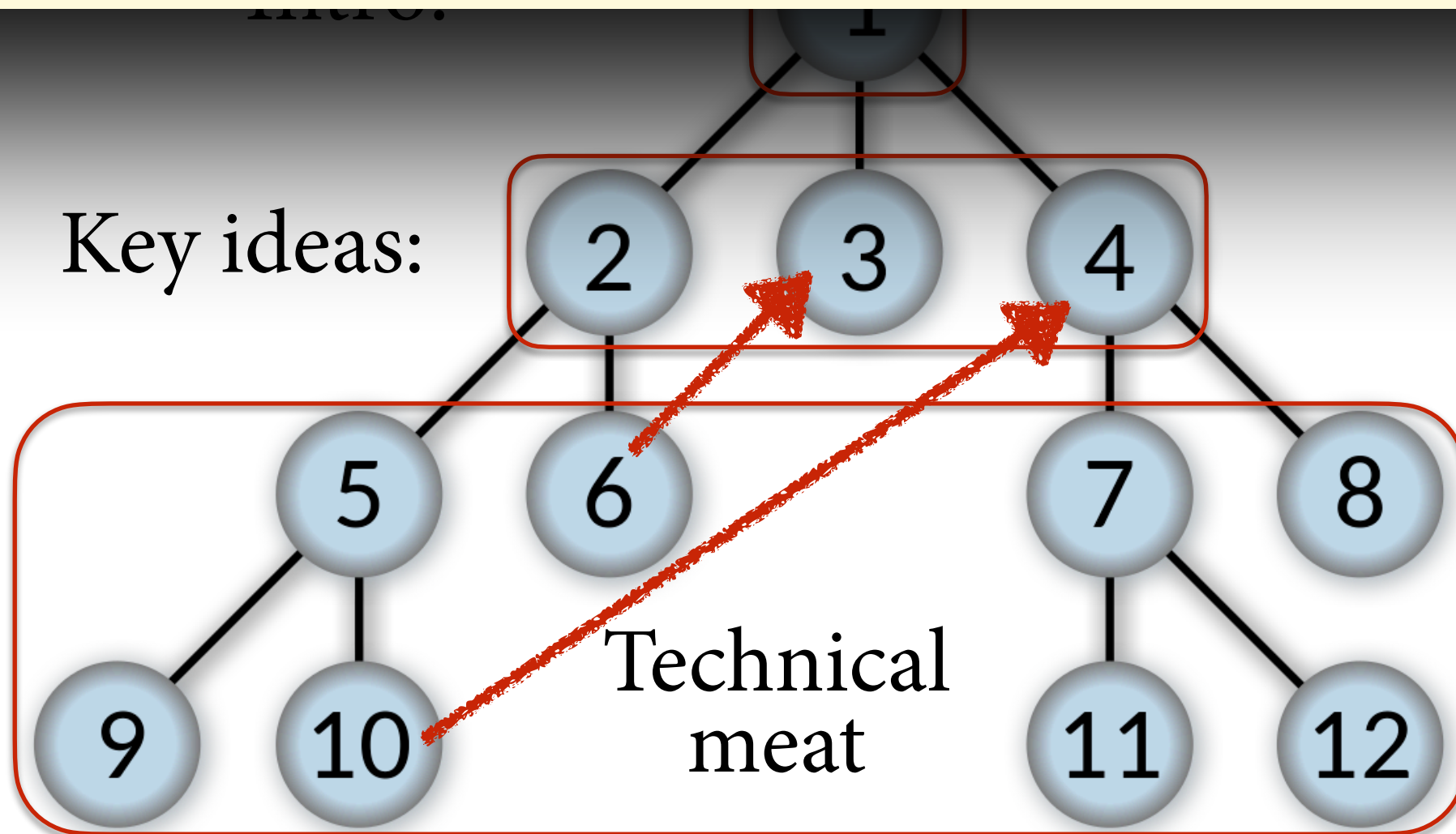
# Breadth-first traversal





# Sometimes breadth-first doesn't work!

e.g., if explaining 3 & 4 requires first explaining subtree rooted at 2





# A Promising Semantics for Relaxed-Memory Concurrency

Jeehoon Kang    Chung-Kil Hur<sup>\*</sup>

Seoul National University, Korea  
 {jeehoon.kang,gil.hur}@sf.snu.ac.kr

Ori Lahav    Viktor Vafeiadis    Derek Dreyer

MPI-SWS, Germany<sup>†</sup>  
 {orilahav,viktor,dreyer}@mpi-sws.org



## Abstract

Despite many years of research, it has proven very difficult to develop a memory model for concurrent programming languages that adequately balances the conflicting desiderata of programmers, compilers, and hardware. In this paper, we propose the first relaxed memory model that (1) accounts for a broad spectrum of features from the C++11 concurrency model, (2) is implementable, in the sense that it provably validates many standard compiler optimizations and reorderings, as well as standard compilation schemes to x86-TSO and Power, (3) justifies simple invariant-based reasoning, thus demonstrating the absence of bad “out-of-thin-air” behaviors, (4) supports “DRF” guarantees, ensuring that programmers who use sufficient synchronization need not understand the full complexities of relaxed-memory semantics, and (5) defines the semantics of racy programs without relying on undefined behaviors, which is a prerequisite for applicability to type-safe languages like Java.

The key novel idea behind our model is the notion of *promises*: a thread may promise to execute a write in the future, thus enabling other threads to read from that write out of order. Crucially, to

memory shared by all threads. To simulate SC semantics on these architectures, one must therefore insert expensive fence instructions to subvert the efforts of the hardware. Secondly, a number of common compiler optimizations—such as constant propagation—are rendered unsound by a naive SC semantics because they effectively reorder memory operations. Moreover, SC semantics is stronger (*i.e.*, more restrictive) than necessary for many concurrent algorithms.

Hence, languages like Java and C++ have opted instead to provide *relaxed* (aka *weak*) memory models [22, 13], which enable programmers to demand SC semantics when they need it, but which also support a range of cheaper memory operations that trade off strongly consistent and/or well-defined behavior for efficiency.

## 1.1 Criteria for a Programming Language Memory Model

Unfortunately, despite many years of research, it has proven very difficult to develop a memory model for concurrent programming languages that adequately balances the conflicting desiderata of programmers, compilers, and hardware. In particular, we would like to find a memory model that satisfies the following properties:

$$\begin{array}{c}
\text{(MEMORY: NEW)} \\
\hline
\langle P, M \rangle \xrightarrow{m} \langle P, M \stackrel{\Delta}{\leftarrow} m \rangle
\end{array}
\quad
\begin{array}{c}
\text{(MEMORY: FULFILL)} \\
\hline
\leftarrow \in \{\stackrel{S}{\leftarrow}, \stackrel{U}{\leftarrow}\} \quad P' = P \leftarrow m \quad M' = M \leftarrow m \\
\hline
\langle P, M \rangle \xrightarrow{m} \langle P' \setminus \{m\}, M' \rangle
\end{array}$$
  

$$\begin{array}{c}
\text{(READ-HELPER)} \\
o = \text{pln} \implies \text{cur.pln}(x) \leq t \\
o \in \{\text{rlx}, \text{ra}\} \implies \text{cur.rlx}(x) \leq t \\
\text{cur}' = \text{cur} \sqcup V \sqcup (o \sqsupseteq \text{ra} ? R) \\
\text{acq}' = \text{acq} \sqcup V \sqcup (o \sqsupseteq \text{rlx} ? R) \\
\text{where } V = [\text{pln} : (o \sqsupseteq \text{rlx} ? \{x@t\}), \text{rlx} : \{x@t\}] \\
\hline
\langle \text{cur}, \text{acq}, \text{rel} \rangle \xrightarrow{R:o,x,t,R} \langle \text{cur}', \text{acq}', \text{rel} \rangle
\end{array}
\quad
\begin{array}{c}
\text{(WRITE-HELPER)} \\
\text{cur.rlx}(x) < t \\
\text{cur}' = \text{cur} \sqcup V \quad \text{acq}' = \text{acq} \sqcup \text{cur}' \\
\text{rel}' = \text{rel}[x \mapsto \text{rel}(x) \sqcup V \sqcup (o \sqsupseteq \text{ra} ? \text{cur}')] \\
R_w = (o \sqsupseteq \text{rlx} ? (\text{rel}'(x) \sqcup R_r)) \\
\text{where } V = [\text{pln} : \{x@t\}, \text{rlx} : \{x@t\}] \\
\hline
\langle \text{cur}, \text{acq}, \text{rel} \rangle \xrightarrow{W:o,x,t,R_r,R_w} \langle \text{cur}', \text{acq}', \text{rel}' \rangle
\end{array}$$
  

$$\begin{array}{c}
\text{(SC-FENCE-HELPER)} \\
\text{S}' = \text{acq.rlx} \sqcup \text{S} \\
\text{cur}' = \text{acq}' = \langle \text{S}', \text{S}' \rangle \\
\text{rel}' = \lambda_. \langle \text{S}', \text{S}' \rangle \\
\hline
\langle \langle \text{cur}, \text{acq}, \text{rel} \rangle, \text{S} \rangle \xrightarrow{\text{F}_{\text{sc}}} \langle \langle \text{cur}', \text{acq}', \text{rel}' \rangle, \text{S}' \rangle
\end{array}$$
  

$$\begin{array}{c}
\text{(UPDATE)} \\
\sigma \xrightarrow{U(o_r, o_w, x, v_r, v_w)} \sigma' \\
o_w = \text{ra} \implies \forall m' \in P(x). m'.\text{view} = \perp \\
\langle x : v_r @(\_, t_r], R_r \rangle \in M \\
m_w = \langle x : v_w @ (t_r, t_w], R_w \rangle \\
\langle P, M \rangle \xrightarrow{m_w} \langle P', M' \rangle \\
\mathcal{V} \xrightarrow{R:o_r,x,t_r,R_r} \mathcal{V}' \xrightarrow{W:o_w,x,t_w,R_r,R_w} \mathcal{V}' \\
\hline
\langle \langle \sigma, \mathcal{V}, P \rangle, \mathcal{S}, M \rangle \rightarrow \langle \langle \sigma', \mathcal{V}', P' \rangle, \mathcal{S}, M' \rangle
\end{array}$$
  

$$\begin{array}{c}
\text{(READ)} \\
\sigma \xrightarrow{R(o,x,v)} \sigma' \\
\langle x : v @(\_, t], R \rangle \in M \\
\mathcal{V} \xrightarrow{R:o,x,t,R} \mathcal{V}' \\
\hline
\langle \langle \sigma, \mathcal{V}, P \rangle, \mathcal{S}, M \rangle \rightarrow \langle \langle \sigma', \mathcal{V}', P \rangle, \mathcal{S}, M \rangle
\end{array}
\quad
\begin{array}{c}
\text{(WRITE)} \\
\sigma \xrightarrow{W(o,x,v)} \sigma' \\
o = \text{ra} \implies \forall m' \in P(x). m'.\text{view} = \perp \\
m = \langle x : v @(\_, t], R \rangle \\
\langle P, M \rangle \xrightarrow{m} \langle P', M' \rangle \\
\mathcal{V} \xrightarrow{W:o,x,t,\perp,R} \mathcal{V}' \\
\hline
\langle \langle \sigma, \mathcal{V}, P \rangle, \mathcal{S}, M \rangle \rightarrow \langle \langle \sigma', \mathcal{V}', P' \rangle, \mathcal{S}, M' \rangle
\end{array}$$
  


$$\begin{array}{c}
\text{(ACQ-FENCE)} \\
\sigma \xrightarrow{\text{F}_{\text{acq}}} \sigma' \quad \text{cur}' = \text{acq} \\
\hline
\langle \langle \sigma, \langle \text{cur}, \text{acq}, \text{rel} \rangle, P \rangle, \mathcal{S}, M \rangle \rightarrow \langle \langle \sigma', \langle \text{cur}', \text{acq}, \text{rel} \rangle, P \rangle, \mathcal{S}, M \rangle
\end{array}
\quad
\begin{array}{c}
\text{(REL-FENCE)} \\
\sigma \xrightarrow{\text{F}_{\text{rel}}} \sigma' \quad \text{rel}' = \lambda_. \text{cur} \\
\forall m \in P. m.\text{view} = \perp \\
\hline
\langle \langle \sigma, \langle \text{cur}, \text{acq}, \text{rel} \rangle, P \rangle, \mathcal{S}, M \rangle \rightarrow \langle \langle \sigma', \langle \text{cur}, \text{acq}, \text{rel}' \rangle, P \rangle, \mathcal{S}, M \rangle
\end{array}$$
  

$$\begin{array}{c}
\text{(SC-FENCE)} \\
\sigma \xrightarrow{\text{F}_{\text{sc}}} \sigma' \\
\langle \mathcal{V}, \mathcal{S} \rangle \xrightarrow{\text{F}_{\text{sc}}} \langle \mathcal{V}', \mathcal{S}' \rangle \\
\forall m \in P. m.\text{view} = \perp \\
\hline
\langle \langle \sigma, \mathcal{V}, P \rangle, \mathcal{S}, M \rangle \rightarrow \langle \langle \sigma', \mathcal{V}', P \rangle, \mathcal{S}', M \rangle
\end{array}
\quad
\begin{array}{c}
\text{(SYSTEM CALL)} \\
\sigma \xrightarrow{\text{SysCall}(v)} \sigma' \\
\langle \mathcal{V}, \mathcal{S} \rangle \xrightarrow{\text{F}_{\text{sc}}} \langle \mathcal{V}', \mathcal{S}' \rangle \\
\forall m \in P. m.\text{view} = \perp \\
\hline
\langle \langle \sigma, \mathcal{V}, P \rangle, \mathcal{S}, M \rangle \xrightarrow{\text{SysCall}(v)} \langle \langle \sigma', \mathcal{V}', P \rangle, \mathcal{S}', M \rangle
\end{array}$$
  

$$\begin{array}{c}
\text{(SILENT)} \\
\sigma \xrightarrow{\text{Silent}} \sigma' \\
\hline
\langle \langle \sigma, \mathcal{V}, P \rangle, \mathcal{S}, M \rangle \rightarrow \langle \langle \sigma', \mathcal{V}, P \rangle, \mathcal{S}, M \rangle
\end{array}
\quad
\begin{array}{c}
\text{(PROMISE)} \\
\leftarrow \in \{\stackrel{\Delta}{\leftarrow}, \stackrel{S}{\leftarrow}, \stackrel{U}{\leftarrow}\} \quad P' = P \leftarrow m \\
M' = M \leftarrow m \quad m.\text{view} \in M' \\
\hline
\langle \langle \sigma, \mathcal{V}, P \rangle, \mathcal{S}, M \rangle \rightarrow \langle \langle \sigma, \mathcal{V}, P' \rangle, \mathcal{S}, M' \rangle
\end{array}$$
  

$$\begin{array}{c}
\text{(MACHINE STEP)} \\
\langle \text{TS}(i), \mathcal{S}, M \rangle \rightarrow^* \langle \text{TS}', \mathcal{S}', M' \rangle \\
\langle \text{TS}', \mathcal{S}', M' \rangle \xrightarrow{e} \langle \text{TS}'', \mathcal{S}'', M'' \rangle \\
\langle \text{TS}'', \mathcal{S}'', M'' \rangle \text{ is consistent} \\
\hline
\langle \text{TS}, \mathcal{S}, M \rangle \xrightarrow{e} \langle \text{TS}[i \mapsto \text{TS}''], \mathcal{S}'', M'' \rangle
\end{array}$$

**Figure 3.** Full operational semantics.

<p>(MEMORY: NEW)</p> $\frac{}{\langle P, M \rangle \xrightarrow{m} \langle P, M \stackrel{A}{\leftarrow} m \rangle}$	<p>(MEMORY: FULFILL)</p> $\frac{\leftarrow \in \{\stackrel{S}{\leftarrow}, \stackrel{U}{\leftarrow}\} \quad P' = P \leftarrow m \quad M' = M \leftarrow m}{\langle P, M \rangle \xrightarrow{m} \langle P' \setminus \{m\}, M' \rangle}$	
<p>(READ-HELPER)</p> $\begin{aligned} o = \text{pln} &\implies \text{cur.pln}(x) \leq t \\ o \in \{\text{rlx}, \text{ra}\} &\implies \text{cur.rlx}(x) \leq t \\ \text{cur}' &= \text{cur} \sqcup V \\ \text{acq}' &= \text{acq} \sqcup V \\ \text{where } V &= [\text{pln} : (o = \text{pln})] \\ \hline \langle \text{cur}, \text{acq}, \text{rel} \rangle &\xrightarrow{\text{R:}} \end{aligned}$	<p>(WRITE-HELPER)</p> $\begin{aligned} \text{cur.rlx}(x) &< t \\ \text{cur}' &= \text{cur} \sqcup V \quad \text{acq}' = \text{acq} \sqcup \text{cur}' \\ \hline \end{aligned}$	<p>(SC-FENCE-HELPER)</p> $\begin{aligned} \text{S}' &= \text{acq.rlx} \sqcup \text{S} \\ \text{r}' &= \text{acq}' = \langle \text{S}', \text{S}' \rangle \\ \text{rel}' &= \lambda \_ . \langle \text{S}', \text{S}' \rangle \\ \hline \langle \text{cur}, \text{acq}, \text{rel} \rangle, \text{S} &\xrightarrow{\text{F}_{\text{sc}}} \langle \text{cur}', \text{acq}', \text{rel}' \rangle, \text{S}' \end{aligned}$
<p>(READ)</p> $\frac{\sigma \xrightarrow{\text{R}(o,x,v)} \sigma' \quad \langle x : v @ (\_ , t], \mathcal{V} \rangle \xrightarrow{\text{R}:o,x,t,\_} \mathcal{V}}{\langle \langle \sigma, \mathcal{V}, P \rangle, \text{S}, M \rangle \rightarrow \langle \sigma', \mathcal{V}, P \rangle, \text{S}, M \rangle}$		$\begin{aligned} \sigma &\xrightarrow{x, v_r, v_w} \sigma' \\ \sigma' &\in P(x). m'. \text{view} = \perp \\ (\_ , t_r], R_r &\in M \\ (t_r, t_w], R_w &\in M \\ \sigma &\xrightarrow{w} \langle P', M' \rangle \\ \sigma &\xrightarrow{o_w, x, t_w, R_r, R_w} \mathcal{V}' \\ \hline \langle \langle \sigma', \mathcal{V}', P' \rangle, \text{S}, M' \rangle &\end{aligned}$
<p>(ACQ-FENCE)</p> $\frac{\sigma \xrightarrow{\text{F}_{\text{acq}}} \sigma' \quad \text{cur}'}{\langle \langle \sigma, \langle \text{cur}, \text{acq}, \text{rel} \rangle, P \rangle, \text{S}, M \rangle \rightarrow \langle \langle \sigma', \langle \text{cur}', \text{acq}, \text{rel} \rangle, P \rangle, \text{S}, M \rangle}$		<p>(SYSTEM CALL)</p> $\begin{aligned} \sigma &\xrightarrow{\text{SysCall}(v)} \sigma' \\ \langle \mathcal{V}, \text{S} \rangle &\xrightarrow{\text{F}_{\text{sc}}} \langle \mathcal{V}', \text{S}' \rangle \\ m \in P. m. \text{view} &= \perp \\ \hline \langle \langle \sigma, \mathcal{V}, P \rangle, \text{S}, M \rangle &\xrightarrow{\text{SysCall}(v)} \langle \langle \sigma', \mathcal{V}', P \rangle, \text{S}', M \rangle \end{aligned}$
<p>(SILENT)</p> $\frac{}{\langle \langle \sigma, \mathcal{V}, P \rangle, \text{S}, M \rangle \rightarrow \langle \langle \sigma', \mathcal{V}, P \rangle, \text{S}, M \rangle}$	<p>(PROMISE)</p> $\frac{\leftarrow \in \{\stackrel{A}{\leftarrow}, \stackrel{S}{\leftarrow}, \stackrel{U}{\leftarrow}\} \quad P' = P \leftarrow m \quad M' = M \leftarrow m \quad m. \text{view} \in M'}{\langle \langle \sigma, \mathcal{V}, P \rangle, \text{S}, M \rangle \rightarrow \langle \langle \sigma, \mathcal{V}, P' \rangle, \text{S}, M' \rangle}$	<p>(MACHINE STEP)</p> $\begin{aligned} \langle \text{TS}(i), \text{S}, M \rangle &\rightarrow^* \langle \text{TS}', \text{S}', M' \rangle \\ \langle \text{TS}', \text{S}', M' \rangle &\xrightarrow{e} \langle \text{TS}'', \text{S}'', M'' \rangle \\ \langle \text{TS}'', \text{S}'', M'' \rangle &\text{ is consistent} \\ \hline \langle \text{TS}, \text{S}, M \rangle &\xrightarrow{e} \langle \text{TS}[i \mapsto \text{TS}''], \text{S}'', M'' \rangle \end{aligned}$

**Figure 3.** Full operational semantics.

# Layering the presentation

$$\begin{array}{c}
 \text{(THREAD: SILENT)} \\
 \frac{\sigma \xrightarrow{\text{Silent}} \sigma'}{\langle \langle \sigma, V, P \rangle, M \rangle \rightarrow \langle \langle \sigma', V, P \rangle, M \rangle} \\
 \\
 \text{(THREAD: PROMISE)} \\
 \frac{M' = M \stackrel{\Delta}{\leftarrow} m \quad P' = P \stackrel{\Delta}{\leftarrow} m}{\langle \langle \sigma, V, P \rangle, M \rangle \rightarrow \langle \langle \sigma, V, P' \rangle, M' \rangle} \\
 \\
 \text{(THREAD: READ)} \\
 \frac{\sigma \xrightarrow{R(x,v)} \sigma' \quad \langle x : v @ t \rangle \in M \quad V(x) \leq t \quad V' = V[x \mapsto t]}{\langle \langle \sigma, V, P \rangle, M \rangle \rightarrow \langle \langle \sigma', V', P \rangle, M \rangle} \\
 \\
 \text{(THREAD: FULFILL)} \\
 \frac{\sigma \xrightarrow{W(x,v)} \sigma' \quad \langle x : v @ t \rangle \in P \quad P' = P \setminus \{ \langle x : v @ t \rangle \} \quad V(x) < t \quad V' = V[x \mapsto t]}{\langle \langle \sigma, V, P \rangle, M \rangle \rightarrow \langle \langle \sigma', V', P' \rangle, M \rangle} \\
 \\
 \text{(THREAD: WRITE)} \\
 \frac{\sigma \xrightarrow{W(x,v)} \sigma' \quad M' = M \stackrel{\Delta}{\leftarrow} \langle x : v @ t \rangle \quad V(x) < t \quad V' = V[x \mapsto t]}{\langle \langle \sigma, V, P \rangle, M \rangle \rightarrow \langle \langle \sigma', V', P \rangle, M' \rangle} \\
 \\
 \text{(MACHINE STEP)} \\
 \frac{\langle \mathcal{TS}(i), M \rangle \rightarrow^+ \langle \mathcal{TS}', M' \rangle \quad \langle \mathcal{TS}', M' \rangle \text{ is consistent}}{\langle \mathcal{TS}, M \rangle \rightarrow \langle \mathcal{TS}[i \mapsto \mathcal{TS}'], M' \rangle}
 \end{array}$$

**Figure 1.** Operational semantics for the simplified model handling only relaxed read and write accesses.

- **Intro**: A few paragraphs about **main key idea**
- **Section 2**: **More details about main key idea** in a simplified version of the semantics
- **Section 3-4**: Presented **other key ideas** and built up to the full semantics **incrementally**

# Layering the presentation

$$\begin{array}{c}
 \text{(THREAD: SILENT)} \\
 \frac{\sigma \xrightarrow{\text{Silent}} \sigma'}{\langle \langle \sigma, V, P \rangle, M \rangle \rightarrow \langle \langle \sigma', V, P \rangle, M \rangle} \\
 \\
 \text{(THREAD: READ)} \\
 \frac{\sigma \xrightarrow{R(x,v)} \sigma' \quad \langle x : v @ t \rangle \in M \quad V(x) \leq t \quad V' = V[x \mapsto t]}{\langle \langle \sigma, V, P \rangle, M \rangle \rightarrow \langle \langle \sigma', V', P \rangle, M \rangle} \\
 \\
 \text{(THREAD: WRITE)} \\
 \frac{\sigma \xrightarrow{W(x,v)} \sigma' \quad M' = M \stackrel{\Delta}{\leftarrow} \langle x : v @ t \rangle \quad V(x) < t \quad V' = V[x \mapsto t]}{\langle \langle \sigma, V, P \rangle, M \rangle \rightarrow \langle \langle \sigma', V', P \rangle, M' \rangle} \\
 \\
 \text{(THREAD: PROMISE)} \\
 \frac{M' = M \stackrel{\Delta}{\leftarrow} \langle x : v @ t \rangle \quad P' = P \setminus \{ \langle x : v @ t \rangle \}}{\langle \langle \sigma, V, P \rangle, M \rangle \rightarrow \langle \langle \sigma, V, P' \rangle, M' \rangle} \\
 \\
 \text{(THREAD: FULFILL)} \\
 \frac{\sigma \xrightarrow{W(x,v)} \sigma' \quad \langle x : v @ t \rangle \in P \quad P' = P \setminus \{ \langle x : v @ t \rangle \}}{\langle \langle \sigma, V, P \rangle, M \rangle \rightarrow \langle \langle \sigma', V, P' \rangle, M \rangle} \\
 \\
 \text{(MACHINE STEP)} \\
 \frac{\langle TS(i), M \rangle \rightarrow^+ \langle TS', M' \rangle}{\langle \langle \sigma, V, P \rangle, M \rangle \rightarrow \langle \langle \sigma, V, P \rangle, M' \rangle}
 \end{array}$$

“The paper is extremely well written.”

“The presentation of the semantics is well-motivated and understandable.”

- Section 3-4: Presented **other key ideas** and built up to the full semantics **incrementally**



# Layering the presentation

$$\begin{array}{c}
 \text{(THREAD: SILENT)} \\
 \frac{\sigma \xrightarrow{\text{Silent}} \sigma'}{\langle \langle \sigma, V, P \rangle, M \rangle \rightarrow \langle \langle \sigma', V, P \rangle, M \rangle} \\
 \\
 \text{(THREAD: PROMISE)} \\
 \frac{M' = M \stackrel{\Delta}{\leftarrow} m \quad P' = P \stackrel{\Delta}{\leftarrow} m}{\langle \langle \sigma, V, P \rangle, M \rangle \rightarrow \langle \langle \sigma, V, P' \rangle, M' \rangle} \\
 \\
 \text{(THREAD: READ)} \\
 \frac{\sigma \xrightarrow{R(x,v)} \sigma' \quad \langle x : v@t \rangle \in M \quad V(x) \leq t \quad V' = V[x \mapsto t]}{\langle \langle \sigma, V, P \rangle, M \rangle \rightarrow \langle \langle \sigma', V', P \rangle, M \rangle} \\
 \\
 \text{(THREAD: FULFILL)} \\
 \frac{\sigma \xrightarrow{W(x,v)} \sigma' \quad \langle x : v@t \rangle \in P \quad P' = P \setminus \{ \langle x : v@t \rangle \} \quad V(x) < t \quad V' = V[x \mapsto t]}{\langle \langle \sigma, V, P \rangle, M \rangle \rightarrow \langle \langle \sigma', V', P' \rangle, M \rangle} \\
 \\
 \text{(THREAD: WRITE)} \\
 \frac{\sigma \xrightarrow{W(x,v)} \sigma' \quad M' = M \stackrel{\Delta}{\leftarrow} \langle x : v@t \rangle \quad V(x) < t \quad V' = V[x \mapsto t]}{\langle \langle \sigma, V, P \rangle, M \rangle \rightarrow \langle \langle \sigma', V', P \rangle, M' \rangle} \\
 \\
 \text{(MACHINE STEP)} \\
 \frac{\langle \mathcal{TS}(i), M \rangle \rightarrow^+ \langle \mathcal{TS}', M' \rangle \quad \langle \mathcal{TS}', M' \rangle \text{ is consistent}}{\langle \mathcal{TS}, M \rangle \rightarrow \langle \mathcal{TS}[i \mapsto \mathcal{TS}'], M' \rangle}
 \end{array}$$

**Figure 1.** Operational semantics for the simplified model handling only relaxed read and write accesses.

- **Intro**: A few paragraphs about **main key idea**
- **Section 2**: **More details about main key idea** in a simplified version of the semantics
- **Section 3-4**: Presented **other key ideas** and built up to the full semantics **incrementally**

# Layering the presentation

$$\begin{array}{c}
 \text{(THREAD: SILENT)} \\
 \frac{\sigma \xrightarrow{\text{Silent}} \sigma'}{\langle \langle \sigma, V, P \rangle, M \rangle \rightarrow \langle \langle \sigma', V, P \rangle, M \rangle} \\
 \\
 \text{(THREAD: PROMISE)} \\
 \frac{M' = M \xleftrightarrow{\Delta} m \quad P' = P \xleftrightarrow{\Delta} m}{\langle \langle \sigma, V, P \rangle, M \rangle \rightarrow \langle \langle \sigma, V, P' \rangle, M' \rangle} \\
 \\
 \text{(THREAD: READ)} \\
 \frac{\sigma \xrightarrow{R(x,v)} \sigma' \quad \langle x : v@t \rangle \in M \quad V(x) \leq t \quad V' = V[x \mapsto t]}{\langle \langle \sigma, V, P \rangle, M \rangle \rightarrow \langle \langle \sigma', V', P \rangle, M \rangle} \\
 \\
 \text{(THREAD: FULFILL)} \\
 \frac{\sigma \xrightarrow{W(x,v)} \sigma' \quad \langle x : v@t \rangle \in P \quad P' = P \setminus \{ \langle x : v@t \rangle \} \quad V(x) < t \quad V' = V[x \mapsto t]}{\langle \langle \sigma, V, P \rangle, M \rangle \rightarrow \langle \langle \sigma', V', P' \rangle, M \rangle} \\
 \\
 \text{(THREAD: WRITE)} \\
 \frac{\sigma \xrightarrow{W(x,v)} \sigma' \quad M' = M \xleftrightarrow{\Delta} \langle x : v@t \rangle \quad V(x) < t \quad V' = V[x \mapsto t]}{\langle \langle \sigma, V, P \rangle, M \rangle \rightarrow \langle \langle \sigma', V', P \rangle, M' \rangle} \\
 \\
 \text{(MACHINE STEP)} \\
 \frac{\langle \mathcal{TS}(i), M \rangle \rightarrow^+ \langle \mathcal{TS}', M' \rangle \quad \langle \mathcal{TS}', M' \rangle \text{ is consistent}}{\langle \mathcal{TS}, M \rangle \rightarrow \langle \mathcal{TS}[i \mapsto \mathcal{TS}'], M' \rangle}
 \end{array}$$

**Figure 1.** Operational semantics for the simplified model handling only relaxed read and write accesses.

- What if you don't have enough space for such a layered presentation?
  - Move some technical details to appendix
  - Submit to a better conference (i.e. a conference with a higher page limit)



# A structure that works

- Abstract (1-2 paragraphs, 1000 readers)
- Intro (2-4 pages, 100 readers)
- Key ideas (4-6 pages, 50 readers)
- Technical meat (8-12 pages, 5 readers)
- **Related work** (1-3 pages, 100 readers)

# Related work

1. **It goes at the end** of the paper.
  - You can only properly compare to related work once you've explained your own.
2. **Give real comparisons**, not a “laundry list”!
  - Explain in detail how your work fills the **G**ap in a way that related work doesn't.

# Summary of principles

- Flow via “old to new”
- Coherence via “one paragraph, one point”
- Name your baby, just in time, short subjects
- CGI model for abstract/intro
- Scaffold presentation with “key ideas” section
- Detailed related work section goes at the end



*That's all Folks!*