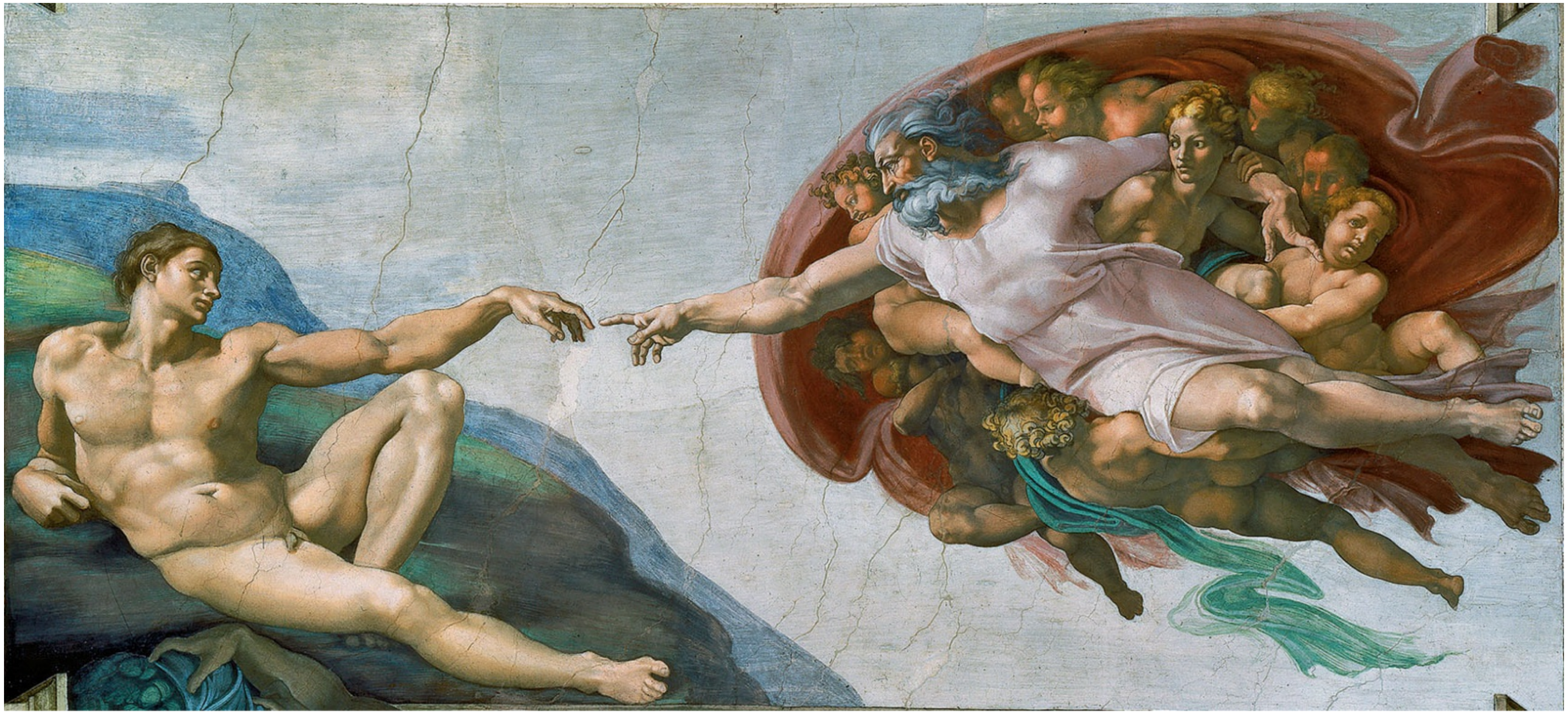
How to Write Papers and Give Talks That People Can Follow



Derek Dreyer

MPI for Software Systems

PLMW@ICFP 2017, Oxford

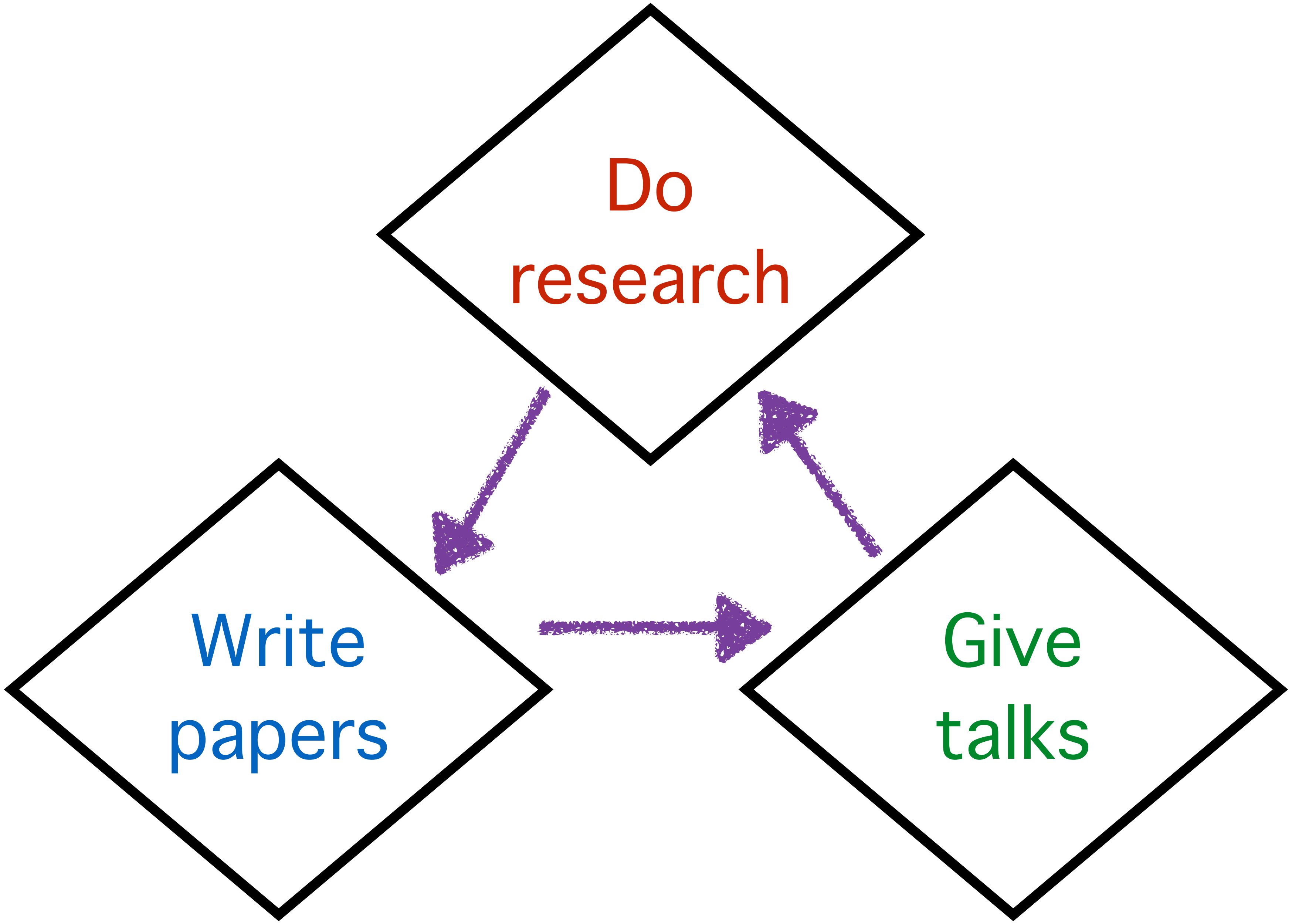
Check out my blog at

herrdreyer.wordpress.com

My job as a researcher

Do research

My job as a researcher

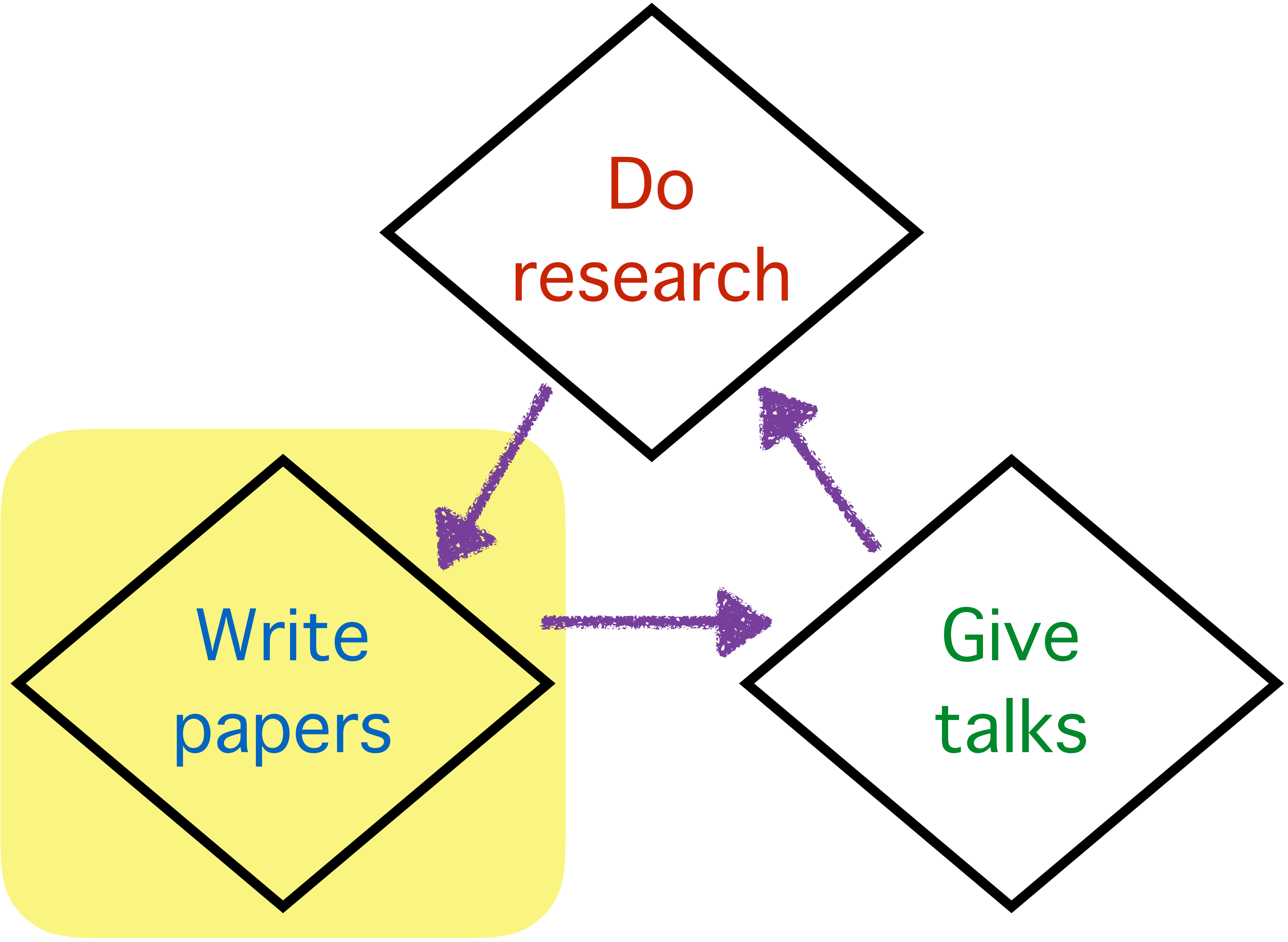


Do research

Write papers

Give talks

My job as a researcher



Have you read any PL papers lately?

Have you read any PL papers lately?

Have you read any PL papers lately?

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

Have you read any PL papers lately?

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

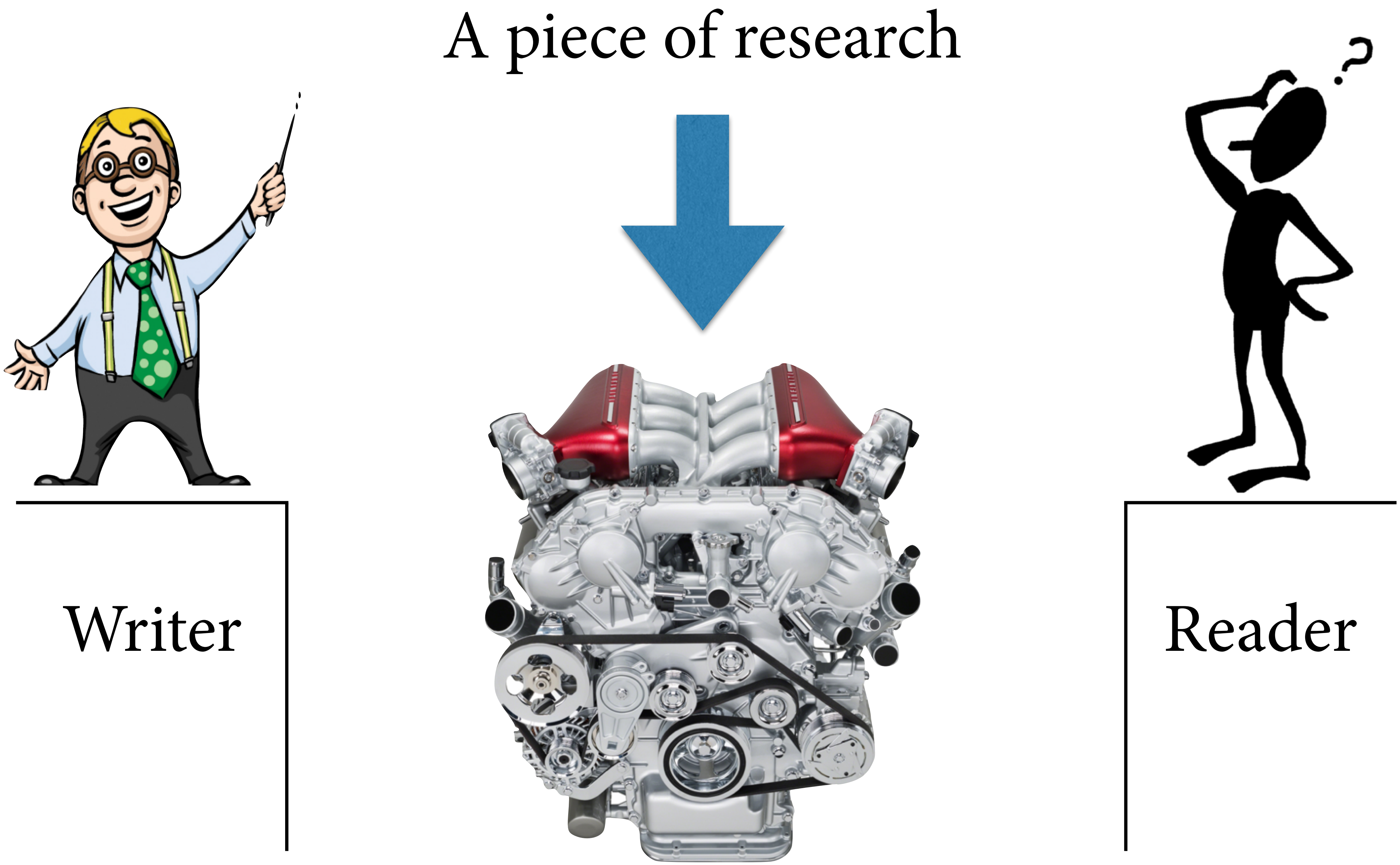
- But in fact, many papers are **poorly written**.

So if you can write clear, accessible papers…

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

So if you can write clear, accessible papers…

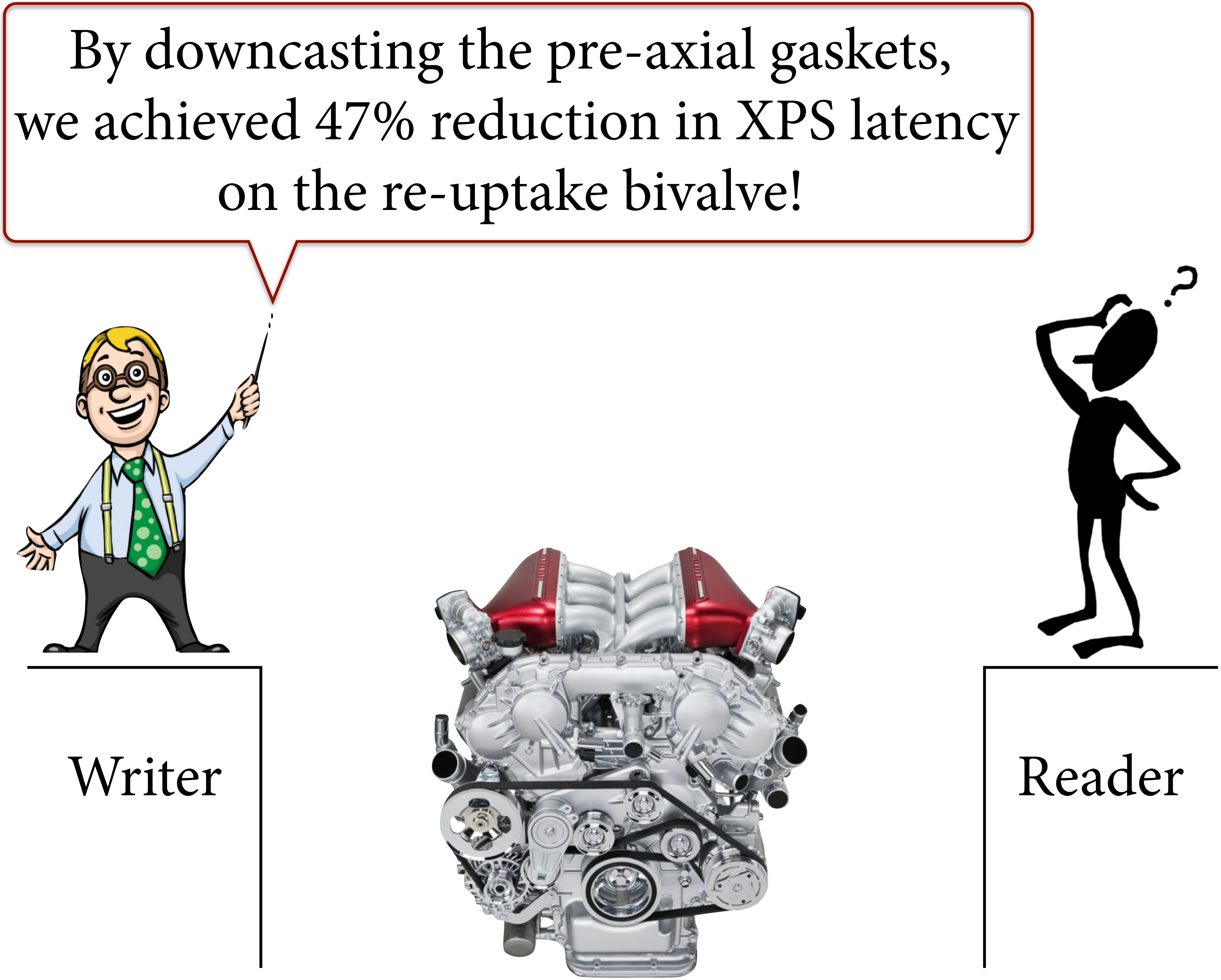
- People will **enjoy** reading them!
- People will **learn** something from them!
- They will get **accepted** to ICFP!

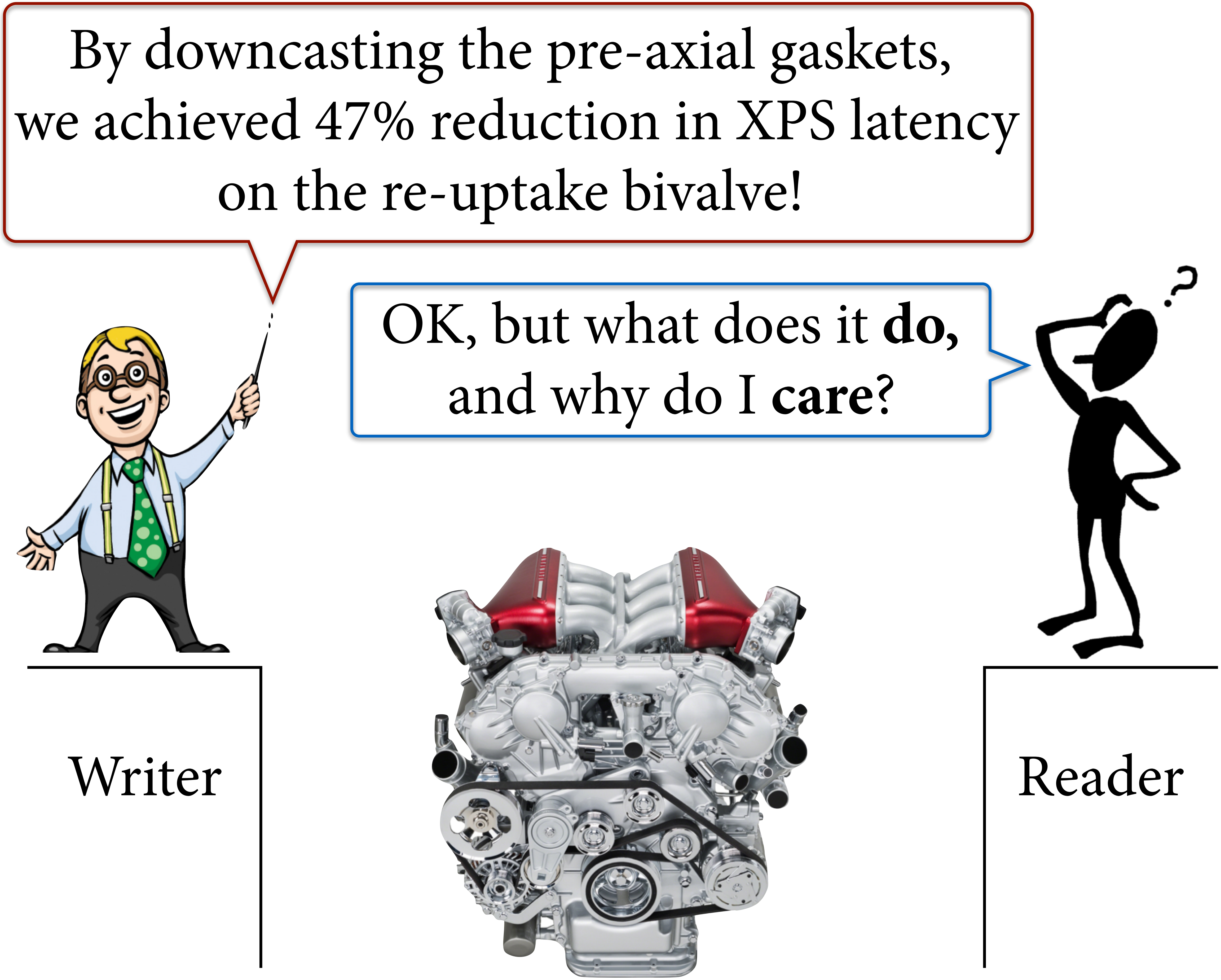


A piece of research

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

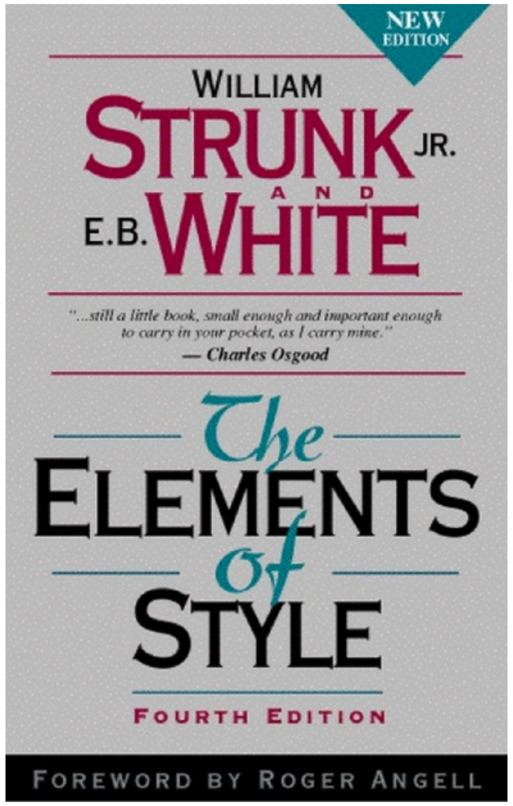
Reader

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

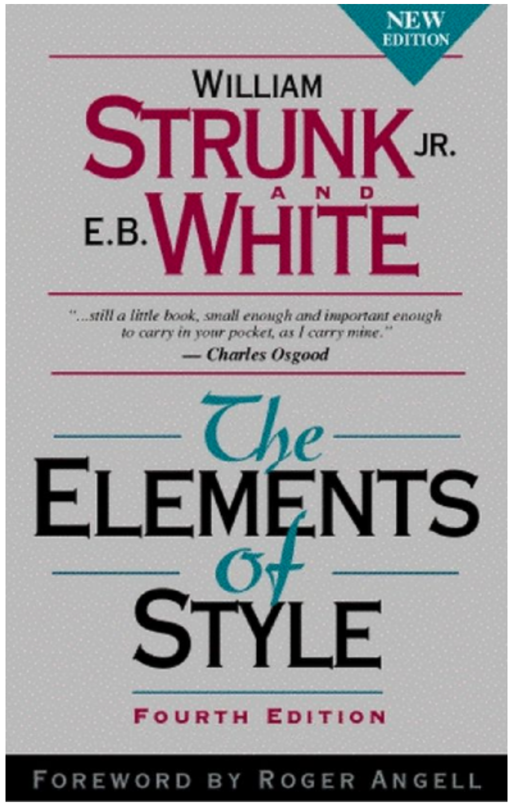
- There are **principles** you can follow that will help you write clearer, more readable prose
  - Based on how readers process information



?

The good news

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



?

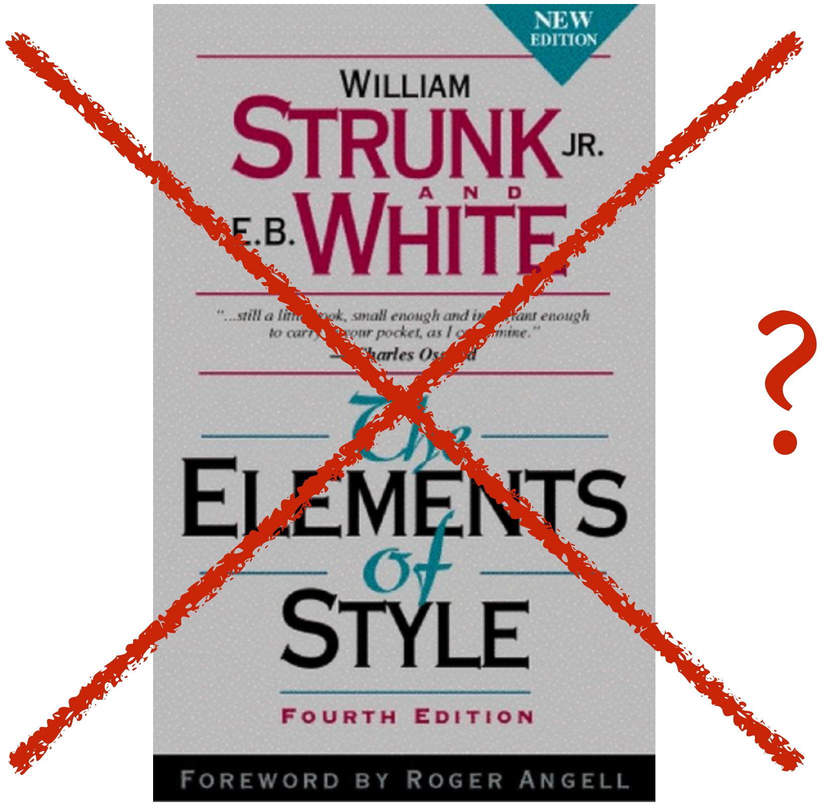
"Be clear"

"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"

…

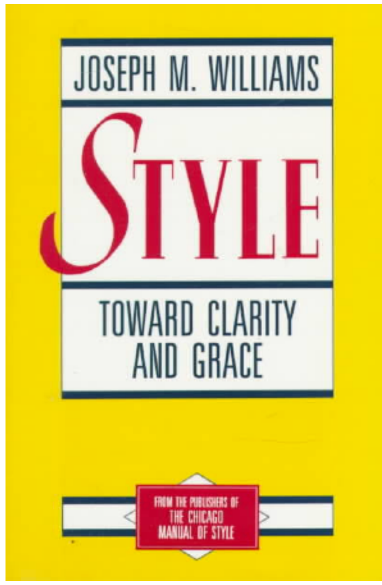
The good news

- 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)

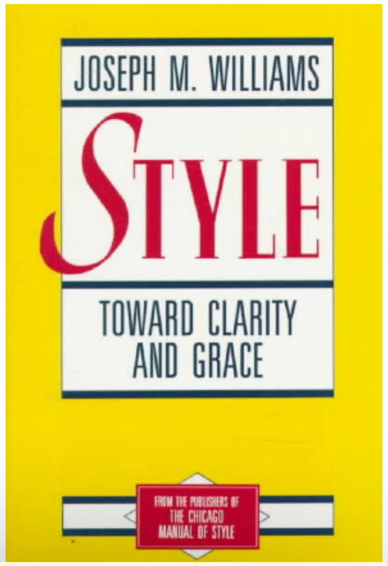


- **Norman Ramsey.** *Learn technical writing in two hours per week.* (course notes)
  - http://www.cs.tufts.edu/~nr/pubs/two.pdf

- **Simon Peyton Jones.** *How to write a great research paper.* (talk)
  - http://research.microsoft.com/en-us/um/people/simonpj/papers/giving-a-talk/giving-a-talk.htm

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)
  - http://research.microsoft.com/en-us/um/people/simonpj/papers/giving-a-talk/giving-a-talk.htm

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

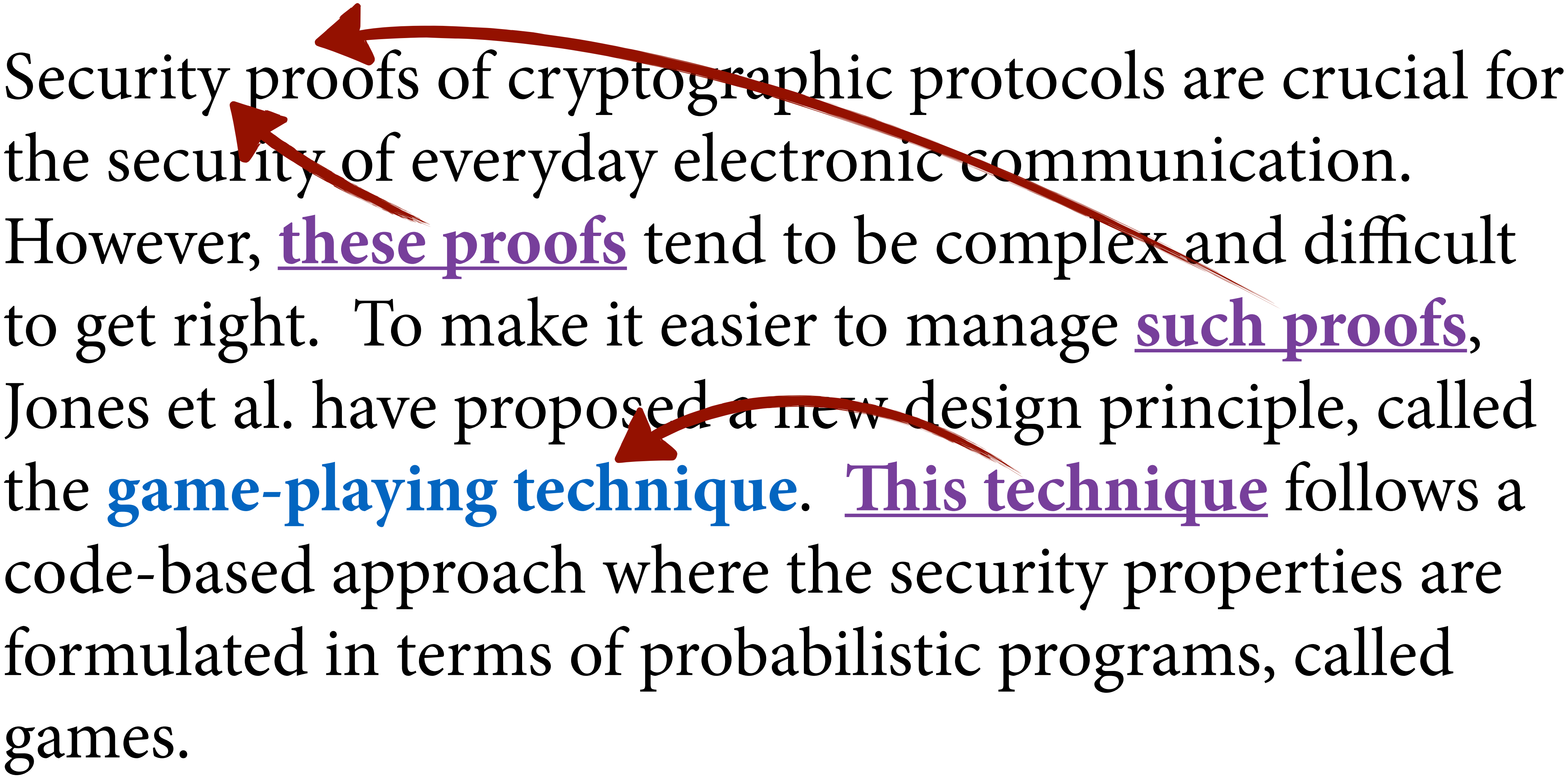
New information

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.

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 mos matic and
awe-inspiring species of c e large
cats, however, n. A
smalle y
successi ats
are curre e world,
they are i ocial. It would
therefore esting 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**

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

Structure of a research paper

The basic idea

TOP-DOWN

Explain your work at multiple levels of abstraction,
starting at a high level and
getting progressively more detailed

A structure that works

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

A structure that works

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

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 Context
  - 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 (1-2 pages, 100 readers)
- **Key ideas** (2-3 pages, 50 readers)
- Technical meat (4-6 pages, 5 readers)
- Related work (1-2 pages, 100 readers)

“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)

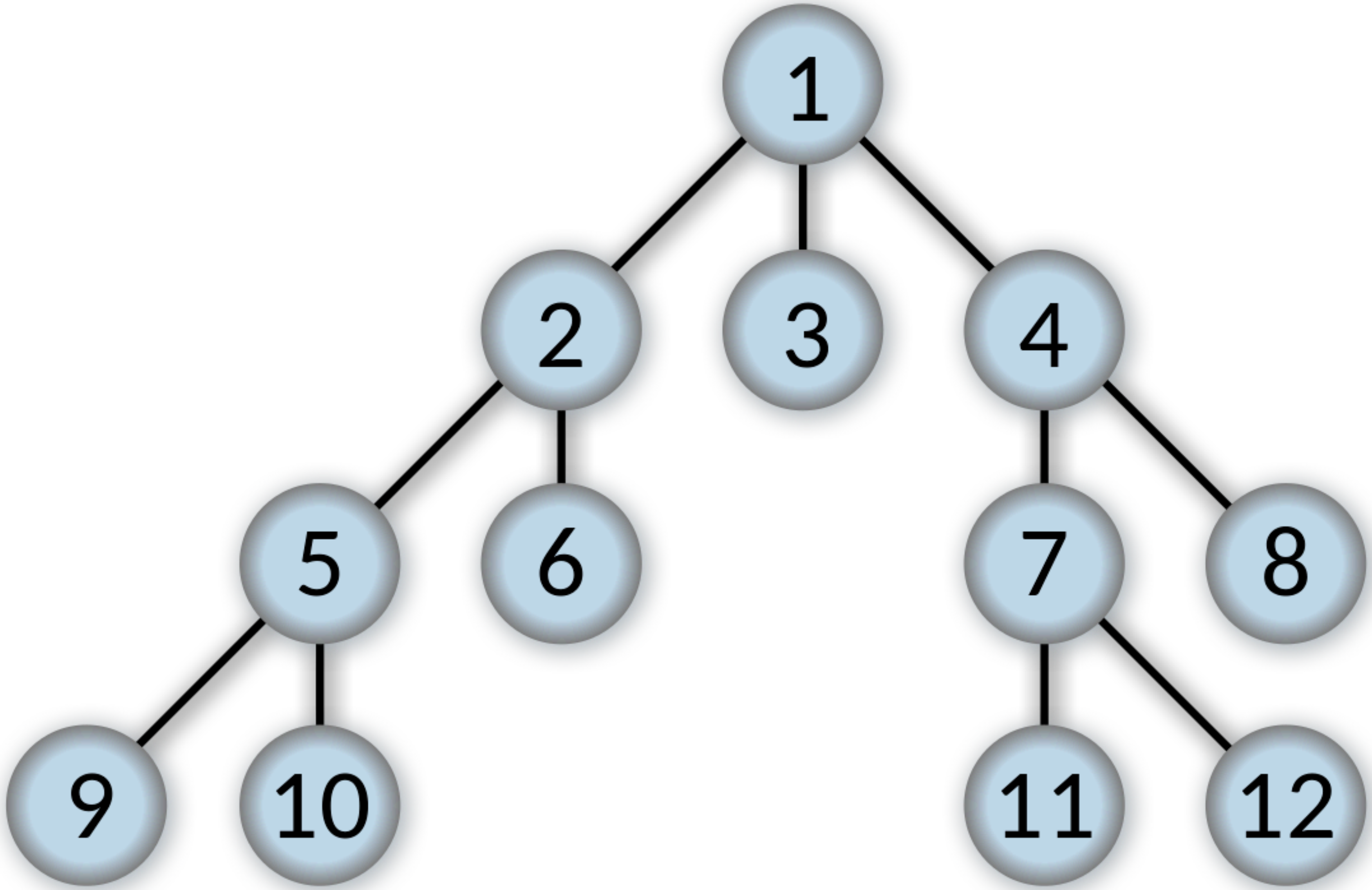
Why have a “key ideas” section at all?

1. Forces you to have a **“takeaway”**
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”

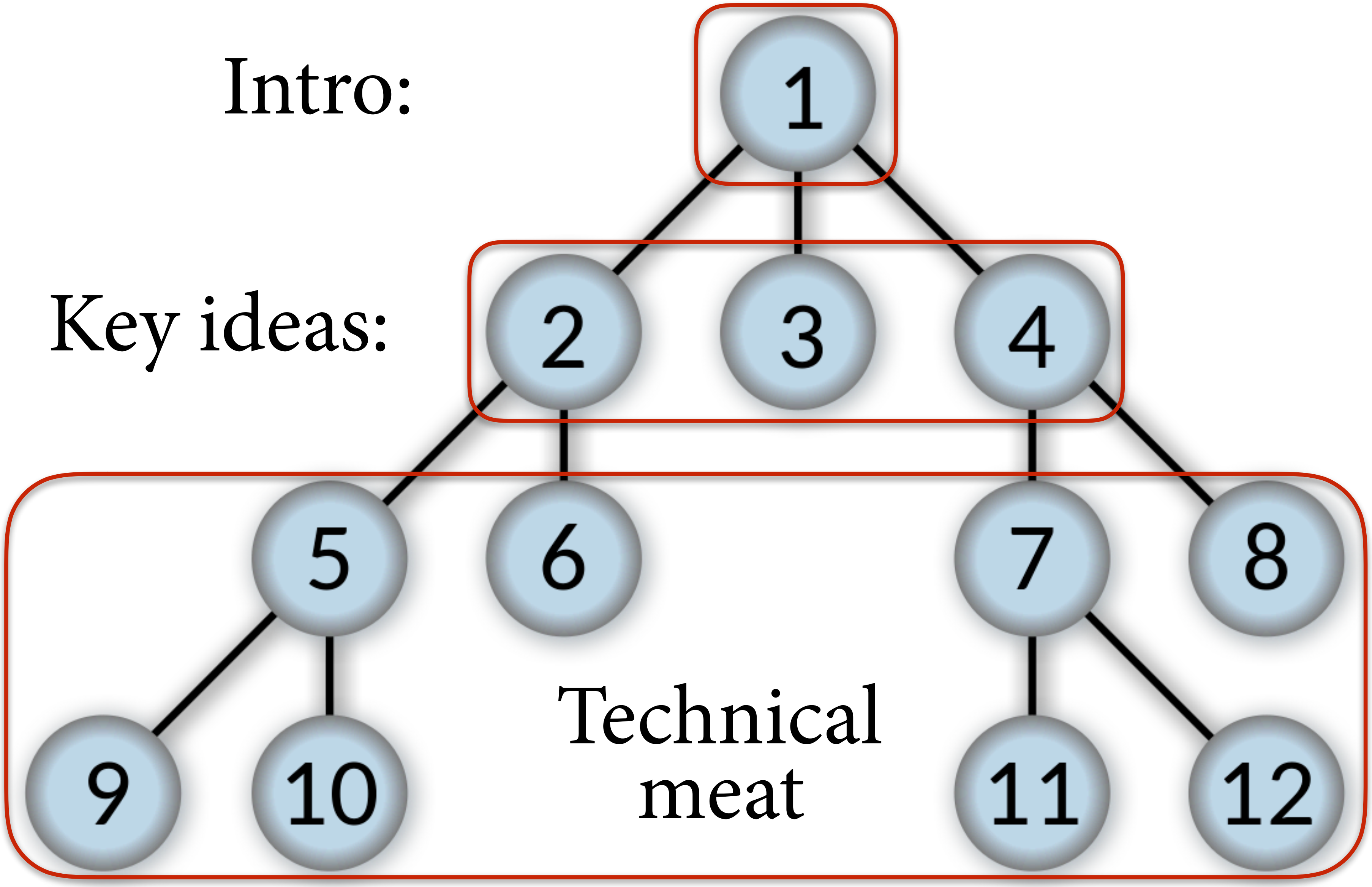
A confession

I don’t always have a key ideas section.

Breadth-first traversal



Breadth-first traversal

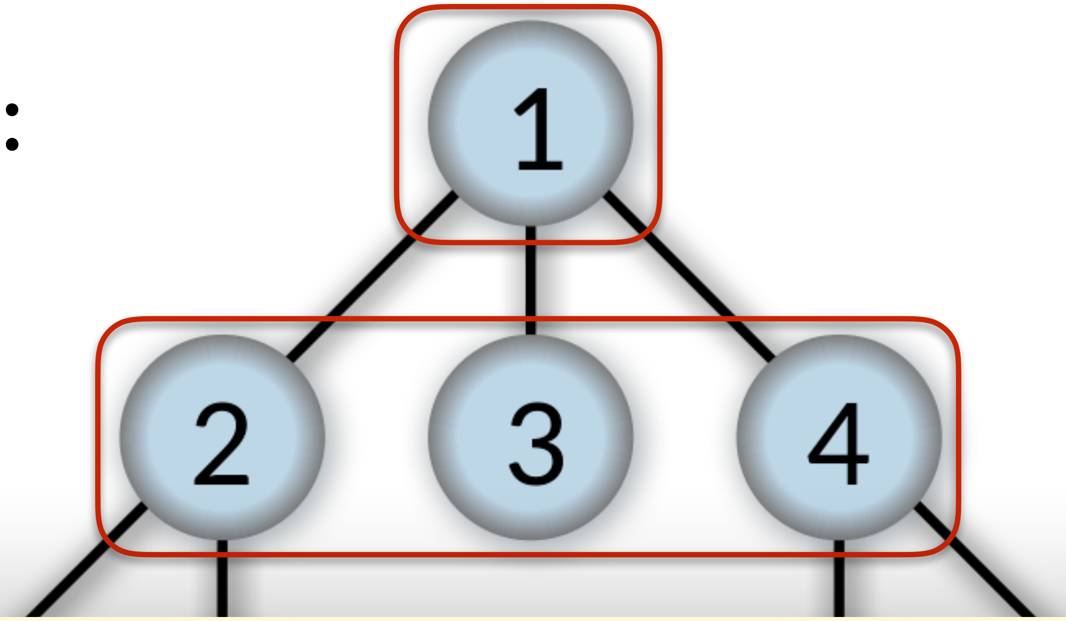


Intro:

Key ideas:

Technical meat

Breadth-first traversal

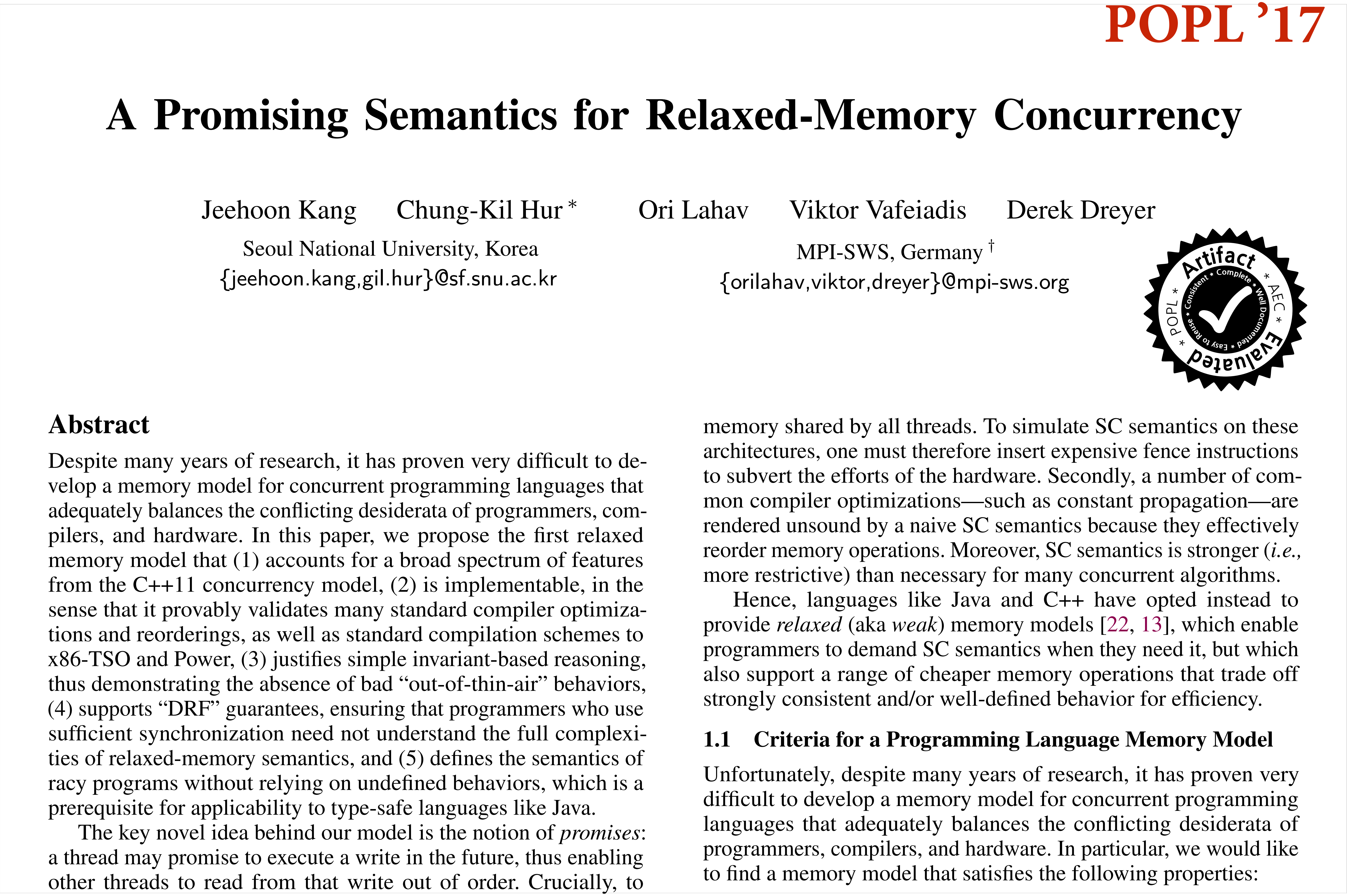


Intro:

Key ideas:

Sometimes breadth-first doesn’t work!

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



POPL ’17

A Promising Semantics for Relaxed-Memory Concurrency

Jeehoon Kang Chung-Kil Hur [*]
Seoul National University, Korea
{jeehoon.kang,gil.hur}@sf.snu.ac.kr

Ori Lahav Viktor Vafeiadis Derek Dreyer
MPI-SWS, Germany [†]
{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:

(MEMORY: NEW)

$$\frac{}{\langle P, M\rangle \xrightarrow{m} \langle P, M \overset{\mathrm{A}}{\hookleftarrow} m\rangle}$$

(MEMORY: FULFILL)

$$\frac{\hookleftarrow \in \{\overset{\mathrm{S}}{\hookleftarrow}, \overset{\mathrm{U}}{\hookleftarrow}\} \qquad P' = P \hookleftarrow m \qquad M' = M \hookleftarrow m}{\langle P, M\rangle \xrightarrow{m} \langle P' \setminus \{m\}, M'\rangle}$$

(READ-HELPER)

$$\frac{\begin{array}{c} o = \mathtt{pln} \quad \Longrightarrow \ cur.\mathtt{pln}(x) \le t \\ o \in \{\mathtt{rlx}, \mathtt{ra}\} \Longrightarrow \ cur.\mathtt{rlx}(x) \le t \\ cur' = cur \sqcup V \sqcup (o \sqsupseteq \mathtt{ra} \ \ ?\, R) \\ acq' = acq \sqcup V \sqcup (o \sqsupseteq \mathtt{rlx}\,?\, R) \\ \textit{where } V = [\mathtt{pln} : (o \sqsupseteq \mathtt{rlx}\,?\, \{x@t\}), \mathtt{rlx} : \{x@t\}] \end{array}}{\langle cur, acq, rel\rangle \xrightarrow{\mathtt{R}:o,x,t,R} \langle cur', acq', rel\rangle}$$

(WRITE-HELPER)

$$\frac{\begin{array}{c} cur.\mathtt{rlx}(x) < t \\ cur' = cur \sqcup V \qquad acq' = acq \sqcup cur' \\ rel' = rel[x \mapsto rel(x) \sqcup V \sqcup (o \sqsupseteq \mathtt{ra}\,?\, cur')] \\ R_{\mathrm{w}} = (o \sqsupseteq \mathtt{rlx}\,?\,(rel'(x) \sqcup R_{\mathrm{r}})) \\ \textit{where } V = [\mathtt{pln} : \{x@t\}, \mathtt{rlx} : \{x@t\}] \end{array}}{\langle cur, acq, rel\rangle \xrightarrow{\mathtt{W}:o,x,t,R_{\mathrm{r}},R_{\mathrm{w}}} \langle cur', acq', rel'\rangle}$$

(SC-FENCE-HELPER)

$$\frac{\begin{array}{c} \mathcal{S}' = acq.\mathtt{rlx} \sqcup \mathcal{S} \\ cur' = acq' = \langle \mathcal{S}', \mathcal{S}'\rangle \\ rel' = \lambda\_.\langle \mathcal{S}', \mathcal{S}'\rangle \end{array}}{\begin{array}{c} \langle\langle cur, acq, rel\rangle, \mathcal{S}\rangle \xrightarrow{\mathtt{F_{sc}}} \\ \langle\langle cur', acq', rel'\rangle, \mathcal{S}'\rangle \end{array}}$$

(READ)

$$\frac{\begin{array}{c} \sigma \xrightarrow{\mathtt{R}(o,x,v)} \sigma' \\ \langle x : v@(\_, t], R\rangle \in M \\ \mathcal{V} \xrightarrow{\mathtt{R}:o,x,t,R} \mathcal{V}' \end{array}}{\langle\langle \sigma, \mathcal{V}, P\rangle, \mathcal{S}, M\rangle \to \langle\langle \sigma', \mathcal{V}', P\rangle, \mathcal{S}, M\rangle}$$

(WRITE)

$$\frac{\begin{array}{c} \sigma \xrightarrow{\mathtt{W}(o,x,v)} \sigma' \\ o = \mathtt{ra} \implies \forall m' \in P(x).\ m'.\mathtt{view} = \bot \\ m = \langle x : v@(\_, t], R\rangle \\ \langle P, M\rangle \xrightarrow{m} \langle P', M'\rangle \\ \mathcal{V} \xrightarrow{\mathtt{W}:o,x,t,\bot,R} \mathcal{V}' \end{array}}{\langle\langle \sigma, \mathcal{V}, P\rangle, \mathcal{S}, M\rangle \to \langle\langle \sigma', \mathcal{V}', P'\rangle, \mathcal{S}, M'\rangle}$$

(UPDATE)

$$\frac{\begin{array}{c} \sigma \xrightarrow{\mathtt{U}(o_{\mathrm{r}},o_{\mathrm{w}},x,v_{\mathrm{r}},v_{\mathrm{w}})} \sigma' \\ o_{\mathrm{w}} = \mathtt{ra} \implies \forall m' \in P(x).\ m'.\mathtt{view} = \bot \\ \langle x : v_{\mathrm{r}} \ @(\_\,, t_{\mathrm{r}}\ ], R_{\mathrm{r}}\ \rangle \in M \\ m_{\mathrm{w}} = \langle x : v_{\mathrm{w}} @(t_{\mathrm{r}}, t_{\mathrm{w}}], R_{\mathrm{w}}\rangle \\ \langle P, M\rangle \xrightarrow{m_{\mathrm{w}}} \langle P', M'\rangle \\ \mathcal{V} \xrightarrow{\mathtt{R}:o_{\mathrm{r}},x,t_{\mathrm{r}},R_{\mathrm{r}}} \xrightarrow{\mathtt{W}:o_{\mathrm{w}},x,t_{\mathrm{w}},R_{\mathrm{r}},R_{\mathrm{w}}} \mathcal{V}' \end{array}}{\langle\langle \sigma, \mathcal{V}, P\rangle, \mathcal{S}, M\rangle \to \langle\langle \sigma', \mathcal{V}', P'\rangle, \mathcal{S}, M'\rangle}$$

(ACQ-FENCE)

$$\frac{\sigma \xrightarrow{\mathtt{F_{acq}}} \sigma' \qquad cur' = acq}{\begin{array}{c} \langle\langle \sigma, \langle cur, acq, rel\rangle, P\rangle, \mathcal{S}, M\rangle \to \\ \langle\langle \sigma', \langle\langle cur', acq, rel\rangle, P\rangle, \mathcal{S}, M\rangle\rangle \end{array}}$$

(REL-FENCE)

$$\frac{\begin{array}{c} \sigma \xrightarrow{\mathtt{F_{rel}}} \sigma' \qquad rel' = \lambda\_.cur \\ \forall m \in P.\ m.\mathtt{view} = \bot \end{array}}{\begin{array}{c} \langle\langle \sigma, \langle cur, acq, rel\rangle, P\rangle, \mathcal{S}, M\rangle \to \\ \langle\langle \sigma', \langle\langle cur, acq, rel'\rangle, P\rangle, \mathcal{S}, M\rangle\rangle \end{array}}$$

(SC-FENCE)

$$\frac{\begin{array}{c} \sigma \xrightarrow{\mathtt{F_{sc}}} \sigma' \\ \langle \mathcal{V}, \mathcal{S}\rangle \xrightarrow{\mathtt{F_{sc}}} \langle \mathcal{V}', \mathcal{S}'\rangle \\ \forall m \in P.\ m.\mathtt{view} = \bot \end{array}}{\begin{array}{c} \langle\langle \sigma, \mathcal{V}, P\rangle, \mathcal{S}, M\rangle \to \\ \langle\langle \sigma', \mathcal{V}', P\rangle, \mathcal{S}', M\rangle \end{array}}$$

(SYSTEM CALL)

$$\frac{\begin{array}{c} \sigma \xrightarrow{\mathrm{SysCall}(v)} \sigma' \\ \langle \mathcal{V}, \mathcal{S}\rangle \xrightarrow{\mathtt{F_{sc}}} \langle \mathcal{V}', \mathcal{S}'\rangle \\ \forall m \in P.\ m.\mathtt{view} = \bot \end{array}}{\begin{array}{c} \langle\langle \sigma, \mathcal{V}, P\rangle, \mathcal{S}, M\rangle \xrightarrow{\mathrm{SysCall}(v)} \\ \langle\langle \sigma', \mathcal{V}', P\rangle, \mathcal{S}', M\rangle \end{array}}$$

(SILENT)

$$\frac{\sigma \xrightarrow{\mathrm{Silent}} \sigma'}{\langle\langle \sigma, \mathcal{V}, P\rangle, \mathcal{S}, M\rangle \to \langle\langle \sigma', \mathcal{V}, P\rangle, \mathcal{S}, M\rangle}$$

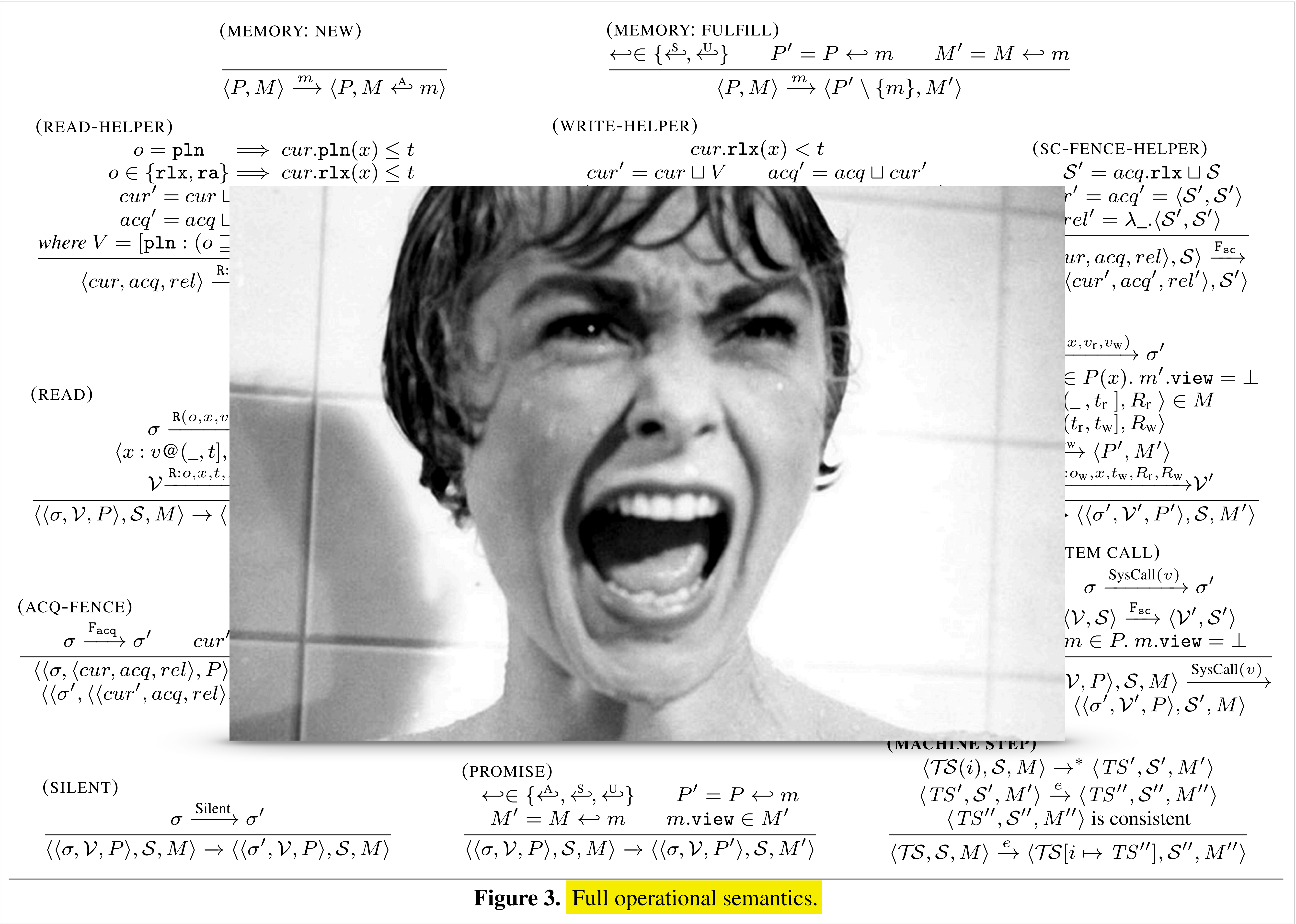
(PROMISE)

$$\frac{\begin{array}{c} \hookleftarrow \in \{\overset{\mathrm{A}}{\hookleftarrow}, \overset{\mathrm{S}}{\hookleftarrow}, \overset{\mathrm{U}}{\hookleftarrow}\} \qquad P' = P \hookleftarrow m \\ M' = M \hookleftarrow m \qquad m.\mathtt{view} \in M' \end{array}}{\langle\langle \sigma, \mathcal{V}, P\rangle, \mathcal{S}, M\rangle \to \langle\langle \sigma, \mathcal{V}, P'\rangle, \mathcal{S}, M'\rangle}$$

(MACHINE STEP)

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

Figure 3. Full operational semantics.



(MEMORY: NEW)

$$\frac{}{\langle P, M\rangle \xrightarrow{m} \langle P, M \overset{\text{A}}{\hookleftarrow} m\rangle}$$

(MEMORY: FULFILL)

$$\frac{\hookleftarrow \in \{\overset{\text{S}}{\hookleftarrow}, \overset{\text{U}}{\hookleftarrow}\} \qquad P' = P \hookleftarrow m \qquad M' = M \hookleftarrow m}{\langle P, M\rangle \xrightarrow{m} \langle P' \setminus \{m\}, M'\rangle}$$

(READ-HELPER)

$o = \texttt{pln} \implies cur.\texttt{pln}(x) \le t$

$o \in \{\texttt{rlx}, \texttt{ra}\} \implies cur.\texttt{rlx}(x) \le t$

$cur' = cur \sqcup$

$acq' = acq \sqcup$

$\textit{where } V = [\texttt{pln} : (o \sqsupseteq$

$\langle cur, acq, rel\rangle \xrightarrow{\texttt{R:}}$

(WRITE-HELPER)

$cur.\texttt{rlx}(x) < t$

$cur' = cur \sqcup V \qquad acq' = acq \sqcup cur'$

(SC-FENCE-HELPER)

$\mathcal{S}' = acq.\texttt{rlx} \sqcup \mathcal{S}$

$r' = acq' = \langle \mathcal{S}', \mathcal{S}'\rangle$

$rel' = \lambda\_.\langle \mathcal{S}', \mathcal{S}'\rangle$

$ur, acq, rel\rangle, \mathcal{S}\rangle \xrightarrow{\texttt{F}_\texttt{sc}}$

$\langle cur', acq', rel'\rangle, \mathcal{S}'\rangle$

(READ)

$\sigma \xrightarrow{\texttt{R}(o, x, v}$

$\langle x : v@(\_, t],$

$\mathcal{V} \xrightarrow{\texttt{R:}o, x, t, }$

$\langle\langle \sigma, \mathcal{V}, P\rangle, \mathcal{S}, M\rangle \rightarrow \langle$

$\xrightarrow{x, v_\text{r}, v_\text{w})} \sigma'$

$\in P(x).\ m'.\texttt{view} = \bot$

$(\_, t_\text{r}], R_\text{r}\rangle \in M$

$(t_\text{r}, t_\text{w}], R_\text{w}\rangle$

$\xrightarrow{\text{w}} \langle P', M'\rangle$

$\xrightarrow{:o_\text{w}, x, t_\text{w}, R_\text{r}, R_\text{w}} \mathcal{V}'$

$\rightarrow \langle\langle \sigma', \mathcal{V}', P'\rangle, \mathcal{S}, M'\rangle$

TEM CALL)

$\sigma \xrightarrow{\text{SysCall}(v)} \sigma'$

$\langle \mathcal{V}, \mathcal{S}\rangle \xrightarrow{\texttt{F}_\texttt{sc}} \langle \mathcal{V}', \mathcal{S}'\rangle$

$m \in P.\ m.\texttt{view} = \bot$

$\mathcal{V}, P\rangle, \mathcal{S}, M\rangle \xrightarrow{\text{SysCall}(v)}$

$\langle\langle \sigma', \mathcal{V}', P\rangle, \mathcal{S}', M\rangle$

(ACQ-FENCE)

$\sigma \xrightarrow{\texttt{F}_\texttt{acq}} \sigma' \qquad cur'$

$\langle\langle \sigma, \langle cur, acq, rel\rangle, P\rangle$

$\langle\langle \sigma', \langle\langle cur', acq, rel\rangle$

(SILENT)

$$\frac{\sigma \xrightarrow{\text{Silent}} \sigma'}{\langle\langle \sigma, \mathcal{V}, P\rangle, \mathcal{S}, M\rangle \rightarrow \langle\langle \sigma', \mathcal{V}, P\rangle, \mathcal{S}, M\rangle}$$

(PROMISE)

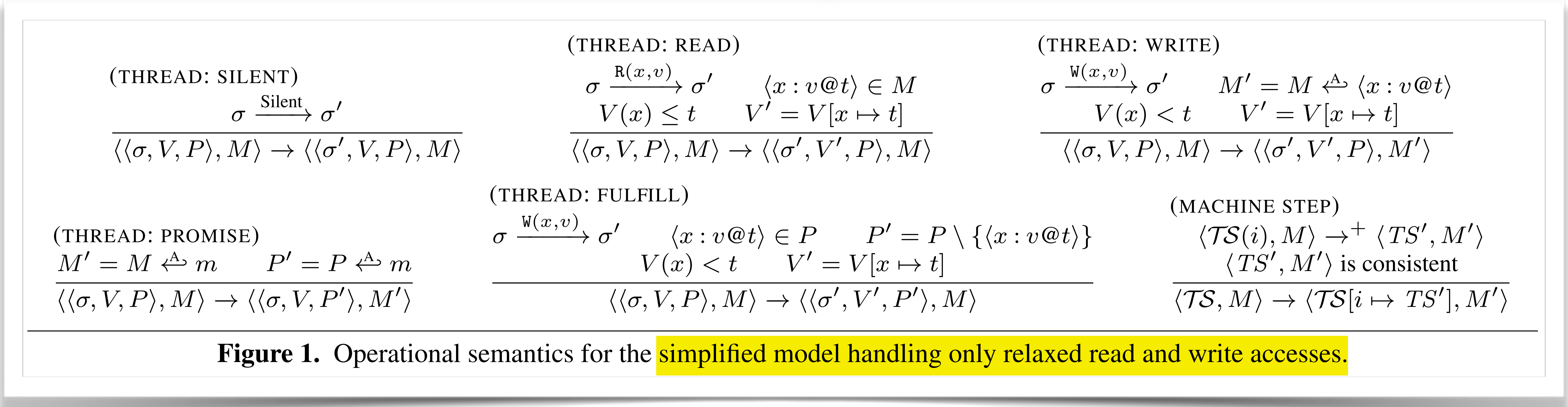
$$\frac{\hookleftarrow \in \{\overset{\text{A}}{\hookleftarrow}, \overset{\text{S}}{\hookleftarrow}, \overset{\text{U}}{\hookleftarrow}\} \qquad P' = P \hookleftarrow m \qquad M' = M \hookleftarrow m \qquad m.\texttt{view} \in M'}{\langle\langle \sigma, \mathcal{V}, P\rangle, \mathcal{S}, M\rangle \rightarrow \langle\langle \sigma, \mathcal{V}, P'\rangle, \mathcal{S}, M'\rangle}$$

(MACHINE STEP)

$$\frac{\langle \mathcal{TS}(i), \mathcal{S}, M\rangle \rightarrow^* \langle TS', \mathcal{S}', M'\rangle \qquad \langle TS', \mathcal{S}', M'\rangle \xrightarrow{e} \langle TS'', \mathcal{S}'', M''\rangle \qquad \langle TS'', \mathcal{S}'', M''\rangle \text{ is consistent}}{\langle \mathcal{TS}, \mathcal{S}, M\rangle \xrightarrow{e} \langle \mathcal{TS}[i \mapsto TS''], \mathcal{S}'', M''\rangle}$$

Figure 3. Full operational semantics.

Layering



(THREAD: SILENT)

$$\frac{\sigma \xrightarrow{\text{Silent}} \sigma'}{\langle\langle \sigma, V, P\rangle, M\rangle \to \langle\langle \sigma', V, P\rangle, M\rangle}$$

(THREAD: READ)

$$\frac{\begin{array}{c}\sigma \xrightarrow{\mathtt{R}(x,v)} \sigma' \qquad \langle x : v@t\rangle \in M \\ V(x) \le t \qquad V' = V[x \mapsto t]\end{array}}{\langle\langle \sigma, V, P\rangle, M\rangle \to \langle\langle \sigma', V', P\rangle, M\rangle}$$

(THREAD: WRITE)

$$\frac{\begin{array}{c}\sigma \xrightarrow{\mathtt{W}(x,v)} \sigma' \qquad M' = M \overset{\text{A}}{\hookleftarrow} \langle x : v@t\rangle \\ V(x) < t \qquad V' = V[x \mapsto t]\end{array}}{\langle\langle \sigma, V, P\rangle, M\rangle \to \langle\langle \sigma', V', P\rangle, M'\rangle}$$

(THREAD: PROMISE)

$$\frac{M' = M \overset{\text{A}}{\hookleftarrow} m \qquad P' = P \overset{\text{A}}{\hookleftarrow} m}{\langle\langle \sigma, V, P\rangle, M\rangle \to \langle\langle \sigma, V, P'\rangle, M'\rangle}$$

(THREAD: FULFILL)

$$\frac{\begin{array}{c}\sigma \xrightarrow{\mathtt{W}(x,v)} \sigma' \qquad \langle x : v@t\rangle \in P \qquad P' = P \setminus \{\langle x : v@t\rangle\} \\ V(x) < t \qquad V' = V[x \mapsto t]\end{array}}{\langle\langle \sigma, V, P\rangle, M\rangle \to \langle\langle \sigma', V', P'\rangle, M\rangle}$$

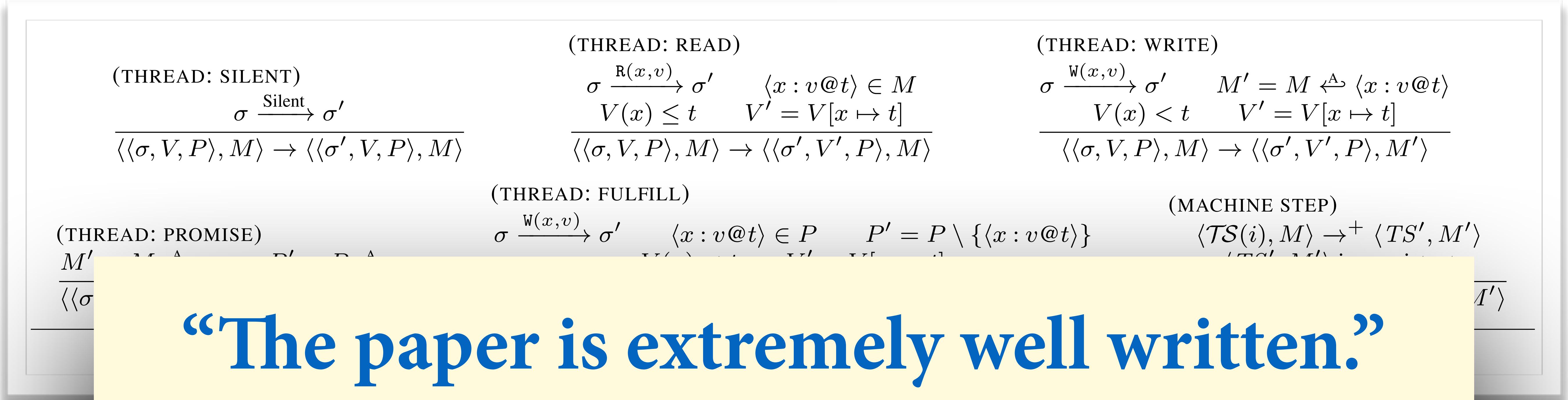
(MACHINE STEP)

$$\frac{\begin{array}{c}\langle \mathcal{TS}(i), M\rangle \to^{+} \langle TS', M'\rangle \\ \langle TS', M'\rangle \text{ is consistent}\end{array}}{\langle \mathcal{TS}, M\rangle \to \langle \mathcal{TS}[i \mapsto TS'], M'\rangle}$$

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

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

Layering



(THREAD: SILENT)

$$\frac{\sigma \xrightarrow{\text{Silent}} \sigma'}{\langle\langle \sigma, V, P\rangle, M\rangle \to \langle\langle \sigma', V, P\rangle, M\rangle}$$

(THREAD: READ)

$$\frac{\sigma \xrightarrow{\mathtt{R}(x,v)} \sigma' \qquad \langle x : v@t\rangle \in M \\ V(x) \le t \qquad V' = V[x \mapsto t]}{\langle\langle \sigma, V, P\rangle, M\rangle \to \langle\langle \sigma', V', P\rangle, M\rangle}$$

(THREAD: WRITE)

$$\frac{\sigma \xrightarrow{\mathtt{W}(x,v)} \sigma' \qquad M' = M \overset{\mathrm{A}}{\hookleftarrow} \langle x : v@t\rangle \\ V(x) < t \qquad V' = V[x \mapsto t]}{\langle\langle \sigma, V, P\rangle, M\rangle \to \langle\langle \sigma', V', P\rangle, M'\rangle}$$

(THREAD: PROMISE)

M'

$\langle\langle\sigma$

(THREAD: FULFILL)

$\sigma \xrightarrow{\mathtt{W}(x,v)} \sigma' \qquad \langle x : v@t\rangle \in P \qquad P' = P \setminus \{\langle x : v@t\rangle\}$

(MACHINE STEP)

$\langle \mathcal{TS}(i), M\rangle \to^{+} \langle TS', M'\rangle$

$M'\rangle$

“The paper is extremely well written.”

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

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

Layering

(THREAD: SILENT)

$$\frac{\sigma \xrightarrow{\text{Silent}} \sigma'}{\langle\langle \sigma, V, P\rangle, M\rangle \to \langle\langle \sigma', V, P\rangle, M\rangle}$$

(THREAD: READ)

$$\frac{\sigma \xrightarrow{\mathtt{R}(x,v)} \sigma' \qquad \langle x : v@t\rangle \in M \qquad V(x) \le t \qquad V' = V[x \mapsto t]}{\langle\langle \sigma, V, P\rangle, M\rangle \to \langle\langle \sigma', V', P\rangle, M\rangle}$$

(THREAD: WRITE)

$$\frac{\sigma \xrightarrow{\mathtt{W}(x,v)} \sigma' \qquad M' = M \overset{\mathsf{A}}{\hookleftarrow} \langle x : v@t\rangle \qquad V(x) < t \qquad V' = V[x \mapsto t]}{\langle\langle \sigma, V, P\rangle, M\rangle \to \langle\langle \sigma', V', P\rangle, M'\rangle}$$

(THREAD: PROMISE)

M'

$\langle\langle\sigma$

NE STEP)

$i), M\rangle \to^{+} \langle TS', M'\rangle$

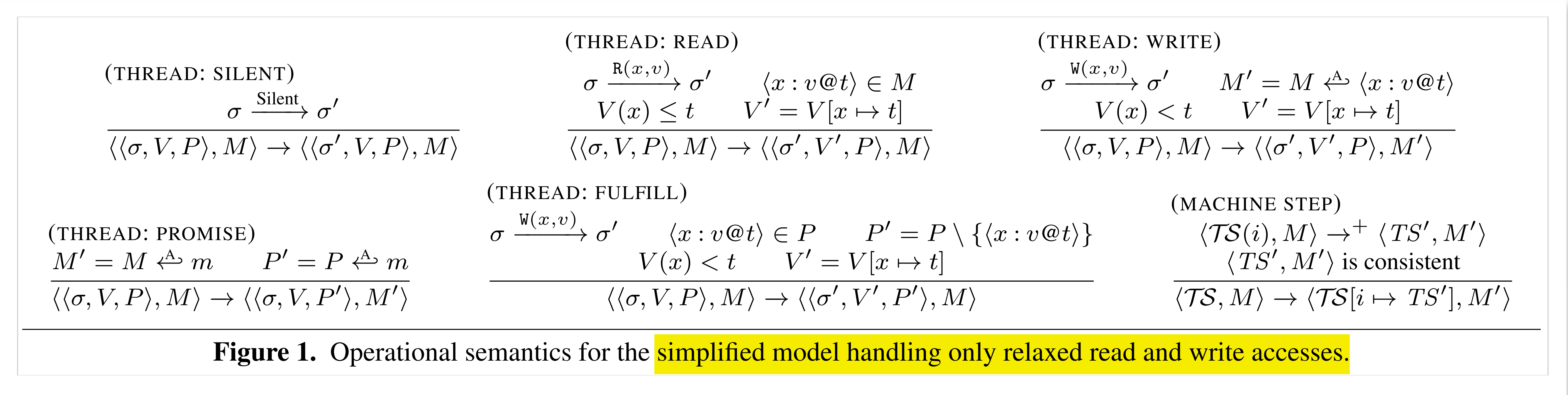
$M'\rangle$

“The ten.”

“The cs is well ble.”

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

Layering



(THREAD: SILENT)

$$\frac{\sigma \xrightarrow{\text{Silent}} \sigma'}{\langle\langle \sigma, V, P\rangle, M\rangle \rightarrow \langle\langle \sigma', V, P\rangle, M\rangle}$$

(THREAD: READ)

$$\frac{\sigma \xrightarrow{\texttt{R}(x,v)} \sigma' \qquad \langle x : v@t\rangle \in M \qquad V(x) \le t \qquad V' = V[x \mapsto t]}{\langle\langle \sigma, V, P\rangle, M\rangle \rightarrow \langle\langle \sigma', V', P\rangle, M\rangle}$$

(THREAD: WRITE)

$$\frac{\sigma \xrightarrow{\texttt{W}(x,v)} \sigma' \qquad M' = M \overset{\scriptscriptstyle A}{\hookleftarrow} \langle x : v@t\rangle \qquad V(x) < t \qquad V' = V[x \mapsto t]}{\langle\langle \sigma, V, P\rangle, M\rangle \rightarrow \langle\langle \sigma', V', P\rangle, M'\rangle}$$

(THREAD: PROMISE)

$$\frac{M' = M \overset{\scriptscriptstyle A}{\hookleftarrow} m \qquad P' = P \overset{\scriptscriptstyle A}{\hookleftarrow} m}{\langle\langle \sigma, V, P\rangle, M\rangle \rightarrow \langle\langle \sigma, V, P'\rangle, M'\rangle}$$

(THREAD: FULFILL)

$$\frac{\sigma \xrightarrow{\texttt{W}(x,v)} \sigma' \qquad \langle x : v@t\rangle \in P \qquad P' = P \setminus \{\langle x : v@t\rangle\} \qquad V(x) < t \qquad V' = V[x \mapsto t]}{\langle\langle \sigma, V, P\rangle, M\rangle \rightarrow \langle\langle \sigma', V', P'\rangle, M\rangle}$$

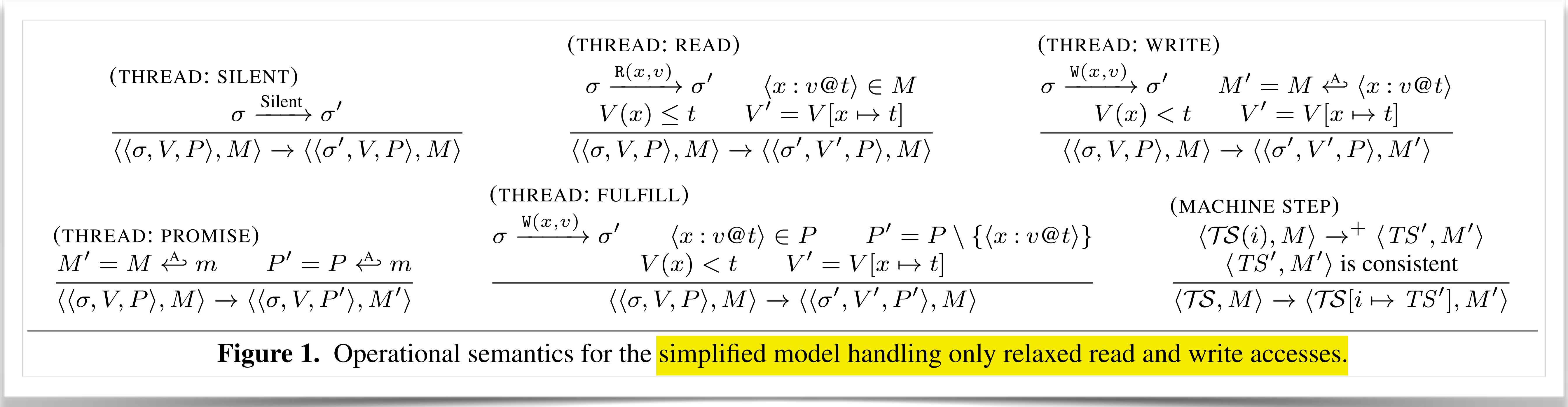
(MACHINE STEP)

$$\frac{\langle \mathcal{TS}(i), M\rangle \rightarrow^{+} \langle \mathcal{TS}', M'\rangle \qquad \langle \mathcal{TS}', M'\rangle \text{ is consistent}}{\langle \mathcal{TS}, M\rangle \rightarrow \langle \mathcal{TS}[i \mapsto \mathcal{TS}'], M'\rangle}$$

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



(THREAD: SILENT)

$$\frac{\sigma \xrightarrow{\text{Silent}} \sigma'}{\langle\langle \sigma, V, P\rangle, M\rangle \to \langle\langle \sigma', V, P\rangle, M\rangle}$$

(THREAD: READ)

$$\frac{\sigma \xrightarrow{\mathtt{R}(x,v)} \sigma' \qquad \langle x : v@t\rangle \in M \\ V(x) \leq t \qquad V' = V[x \mapsto t]}{\langle\langle \sigma, V, P\rangle, M\rangle \to \langle\langle \sigma', V', P\rangle, M\rangle}$$

(THREAD: WRITE)

$$\frac{\sigma \xrightarrow{\mathtt{W}(x,v)} \sigma' \qquad M' = M \overset{A}{\hookleftarrow} \langle x : v@t\rangle \\ V(x) < t \qquad V' = V[x \mapsto t]}{\langle\langle \sigma, V, P\rangle, M\rangle \to \langle\langle \sigma', V', P\rangle, M'\rangle}$$

(THREAD: PROMISE)

$$\frac{M' = M \overset{A}{\hookleftarrow} m \qquad P' = P \overset{A}{\hookleftarrow} m}{\langle\langle \sigma, V, P\rangle, M\rangle \to \langle\langle \sigma, V, P'\rangle, M'\rangle}$$

(THREAD: FULFILL)

$$\frac{\sigma \xrightarrow{\mathtt{W}(x,v)} \sigma' \qquad \langle x : v@t\rangle \in P \qquad P' = P \setminus \{\langle x : v@t\rangle\} \\ V(x) < t \qquad V' = V[x \mapsto t]}{\langle\langle \sigma, V, P\rangle, M\rangle \to \langle\langle \sigma', V', P'\rangle, M\rangle}$$

(MACHINE STEP)

$$\frac{\langle \mathcal{TS}(i), M\rangle \to^{+} \langle TS', M'\rangle \\ \langle TS', M'\rangle \text{ is consistent}}{\langle \mathcal{TS}, M\rangle \to \langle \mathcal{TS}[i \mapsto TS'], M'\rangle}$$

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 (1-2 pages, 100 readers)
- Key ideas (2-3 pages, 50 readers)
- Technical meat (4-6 pages, 5 readers)
- **Related work** (1-2 pages, 100 readers)

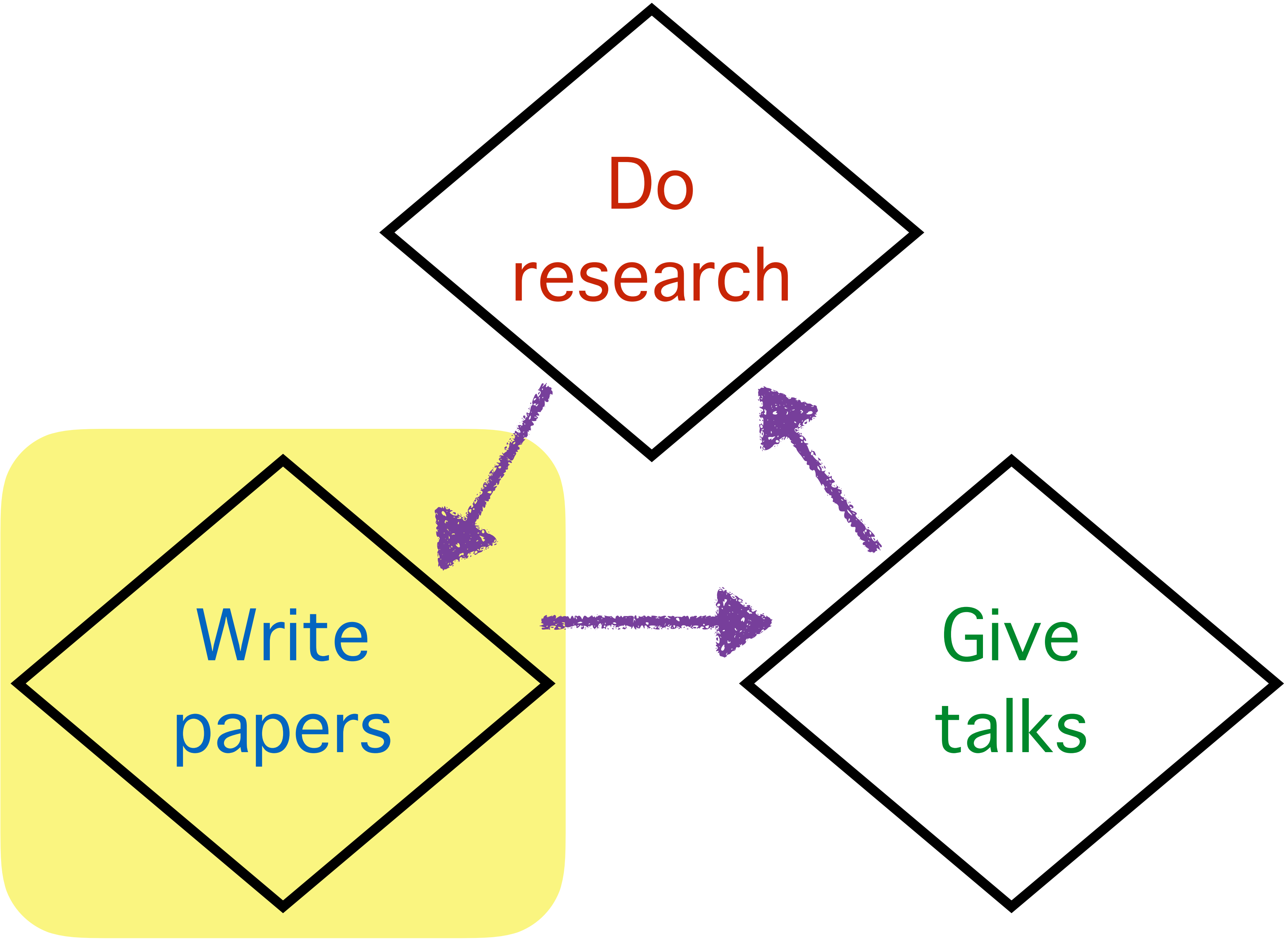
Related work

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.

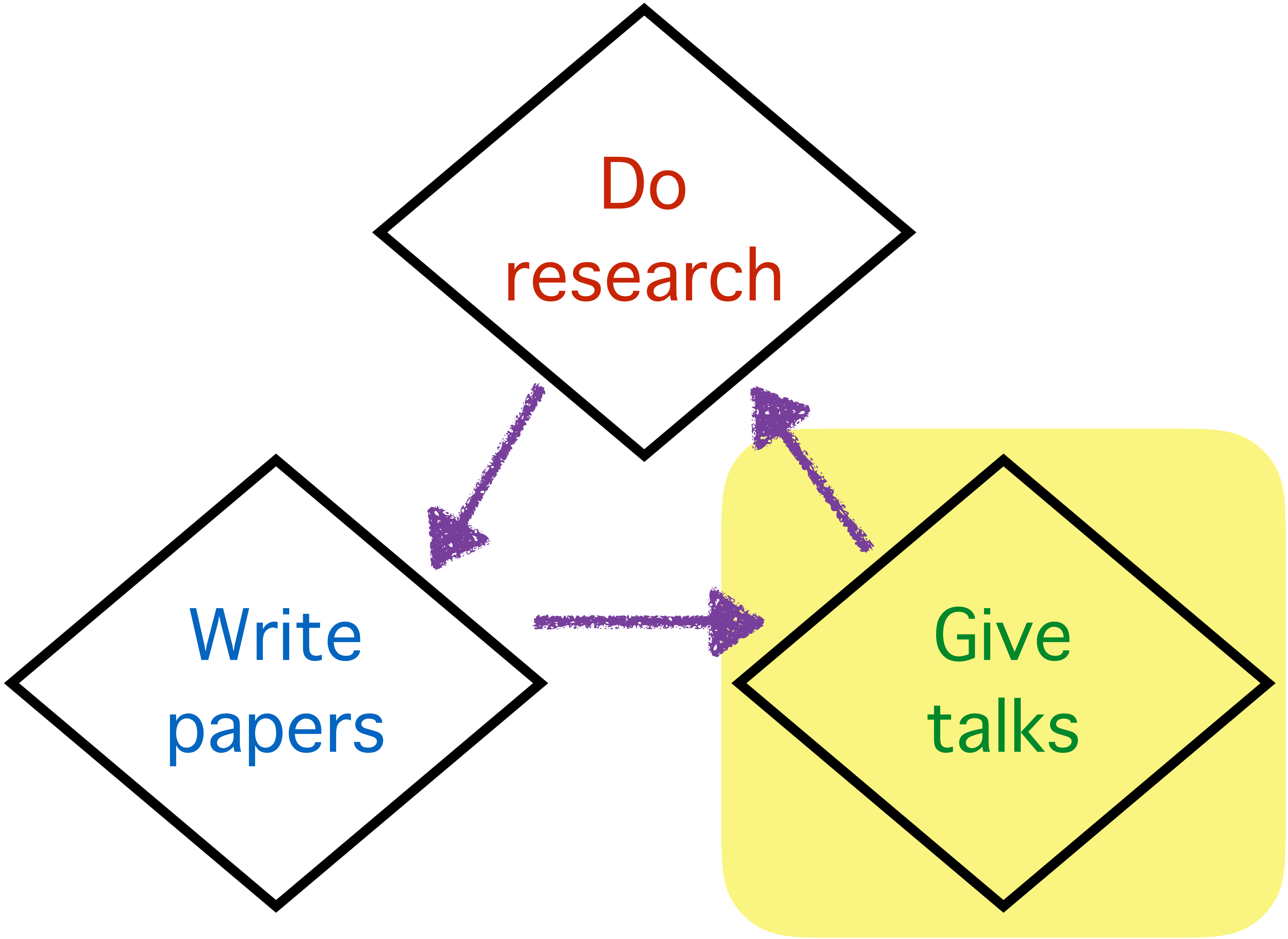
A structure that works

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

My job as a researcher



My job as a researcher



MIZZOU

Entertain your audience!

- **Simon Peyton Jones.** *How to give a great research talk*. (MSR Summer School, 2016)
  - "Your mission is to **wake them up**!"
  - "Your most potent weapon, by far, is **your enthusiasm**!"

- **John Hughes.** *Unaccustomed as I am to public speaking*. (PLMW, 2016)
  - "**Put on a show**!"

Entertain your audience!

- **Simon Peyton Jones.** *How to give a great research talk*. (MSR Summer School, 2016)

Good advice, but I don’t know how to teach people to be entertaining…

- **John Hughes.** *Unaccustomed as I am to public speaking*. (PLMW, 2016)
  - “**Put on a show**!”

How is a conference talk different from a paper?

Conference talks

On the plus side:

✓ Lots of eyeballs on you and your work!

On the minus side:

Conference talks

On the plus side:

✓ Lots of eyeballs on you and your work!

On the minus side:

✗ You can’t say much.

✗ The audience may or may not care.

✗ Even those who care will easily get lost.

Conference talks

On the plus side:

✓ Lots of eyeballs on you and your work!

On the minus side:

✗ You can't say much.

✗ The audience may or may not care.

✗ Even those who care will easily get lost.

A paper structure that works

- **Abstract**
- **Intro**
- **Key ideas**
- **Technical meat**
- **Related work**

A paper structure that works

- **Abstract**
- **Intro**
- **Key ideas**
- **Technical meat**
- **Related work**

A ~~paper~~ talk structure that works

- ~~Abstract~~
- Intro
- Key ideas
- ~~Technical meat~~
- ~~Related work~~

A ~~paper~~ talk structure that works

- ~~Abstract~~
- Intro
- Key ideas
- ~~Technical meat~~
- ~~Related work~~

A ~~paper~~ talk structure that works

- **Intro** (8 minutes)
- **Key ideas** (11 minutes)

talk
A ~~paper~~ structure that works

- **Intro** (8 minutes)
- **Key ideas** (11 minutes)
- **What else is in the paper** (1 minute)

Conference talks

On the plus side:

✓ Lots of eyeballs on you and your work!

On the minus side:

✗ You can’t say much.

✗ The audience may or may not care.

✗ Even those who care will easily get lost.

Conference talks

On the plus side:

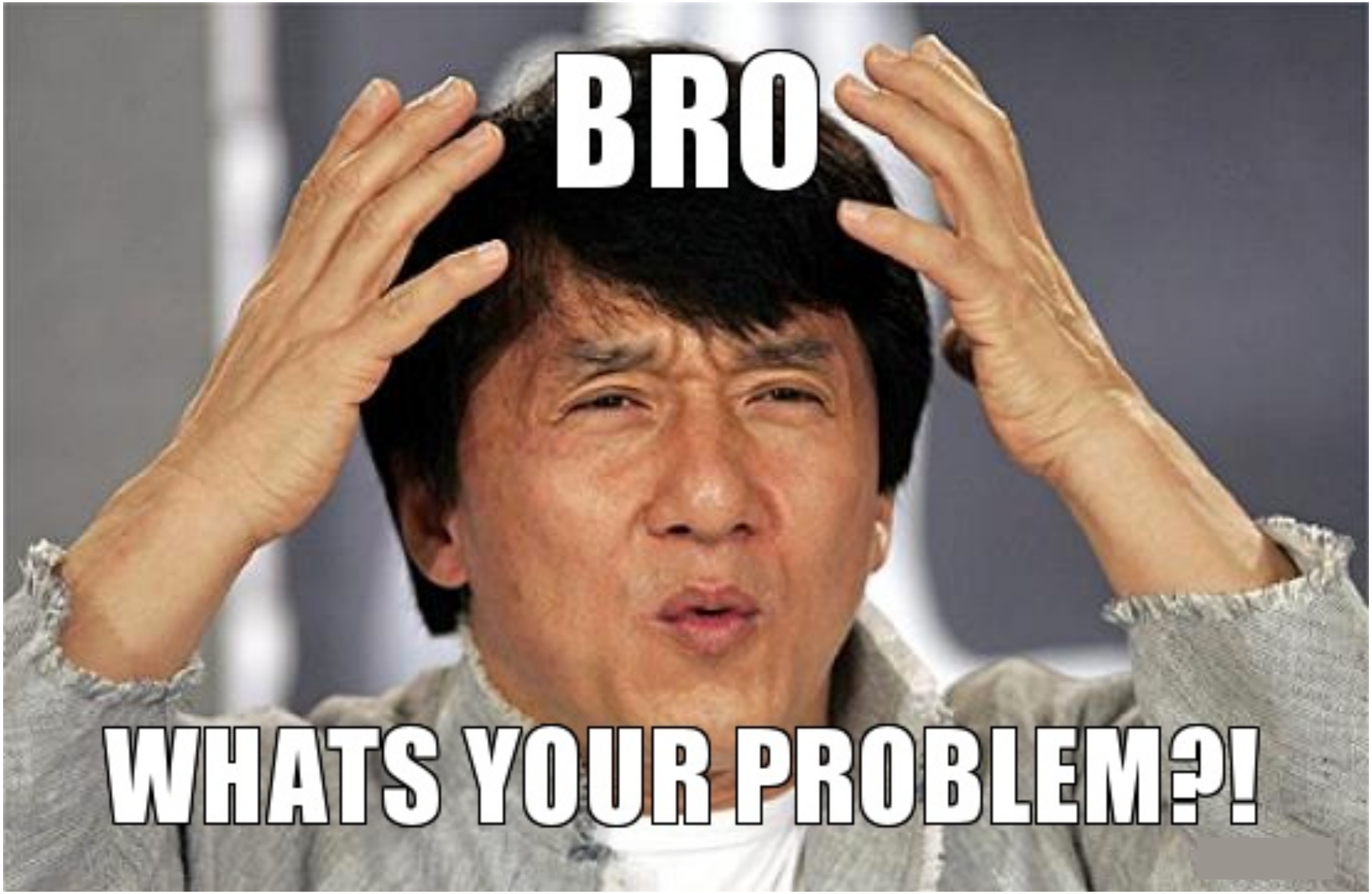
✓ Lots of eyeballs on you and your work!

On the minus side:

✗ You can't say much.

✗ The audience may or may not care.

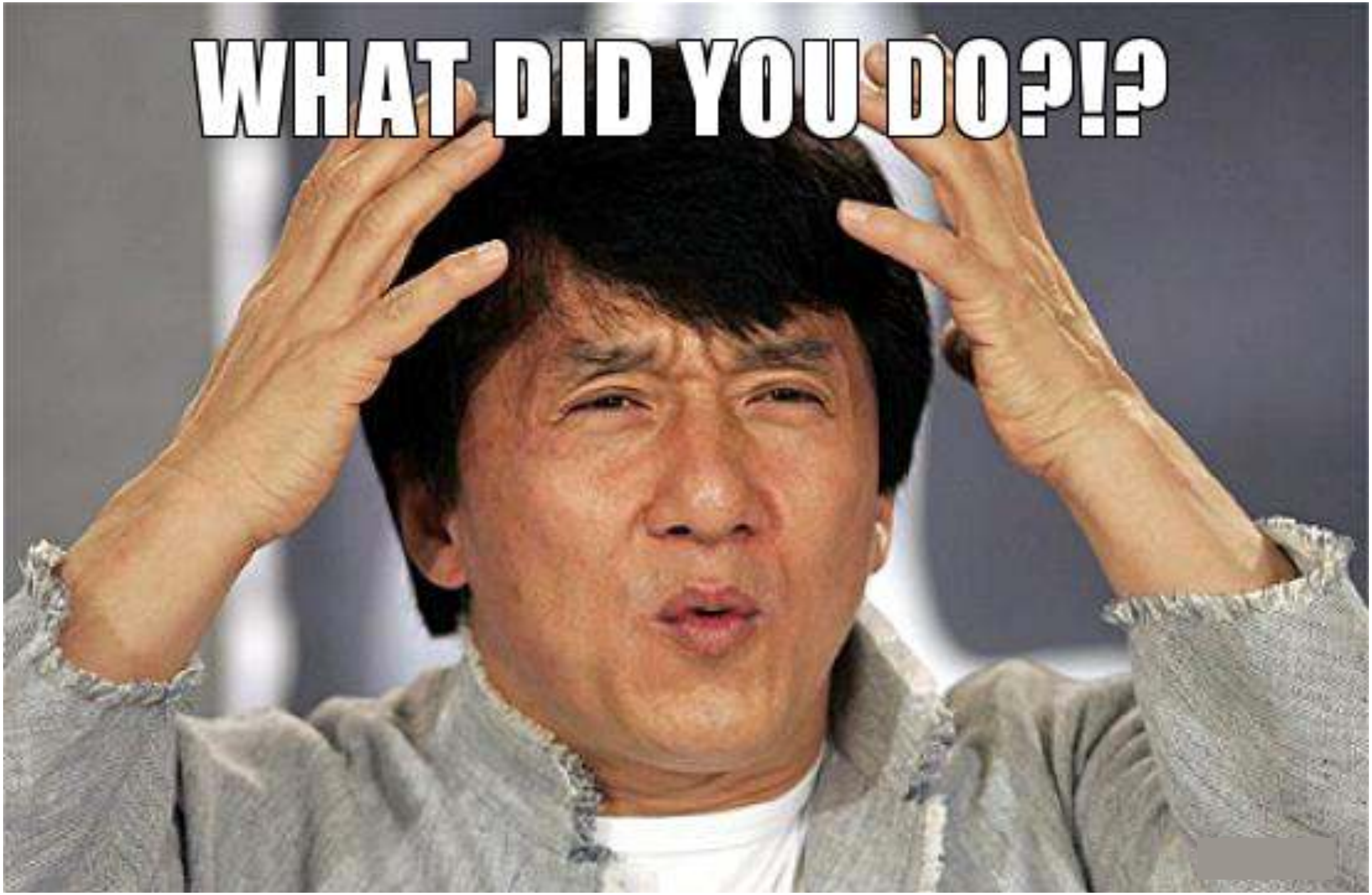
✗ Even those who care will easily get lost.


BRO
WHATS YOUR PROBLEM?!

Stage the motivation

- **First, get to a problem.**
  - Explain a **general** version of your problem (but not too general) **in the first 2 minutes**.

- **Then, get to the problem.**
  - Motivate and **explicitly state** your **specific** problem in the next 4 minutes.
  - Limit discussion of prior work only to what is needed to explain your problem.


WHAT DID YOU DO?!?

Tell them what you did!

- **Proudly state your contributions.**
  - After the motivation, the audience eagerly wants to hear what you did. Tell them!

- **Follow immediately with a crisp statement of your key idea(s).**
  - It will give audience a take-home message, and give focus to the rest of your talk.

Conference talks

On the plus side:

✓ Lots of eyeballs on you and your work!

On the minus side:

✗ You can’t say much.

✗ The audience may or may not care.

✗ Even those who care will easily get lost.

Conference talks

On the plus side:

✓ Lots of eyeballs on you and your work!

On the minus side:

✗ You can’t say much.

✗ The audience may or may not care.

✗ Even those who care will easily get lost.

Flow & coherence

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

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

Flow in talks

- **Within** a slide:
  - Script should follow "old to new"
- **Between** slides:
  - Don't just flip to next slide and say, "So…"
  - Plan something to say **during** the transition

Flow & coherence

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

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

Flow & coherence

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

Create **coherence** with

one ~~paragraph~~, one point

slide

Optimization & Concurrency

- Compiler performs several optimizations to generate optimized code.
  - >100 optimizations in GCC, LLVM.

Correct optimizations for sequential programs may be incorrect for shared memory concurrency.

State-of-the-Art:

- Compilers are over-conservative;

  * optimization opportunities are lost.

or

- Buggy optimization

  * *"Premature optimization is the root of all evil"* ~ Donald Knuth

Talklets

- **Break long stretches of talk into talklets.**
  - More digestible units of story (2-4 min.)
  - But just having talklets is not enough…
- **Use transitions between talklets to remind the audience of the big picture.**
  - Summarize the point of the last talklet and how it connects to the next one.

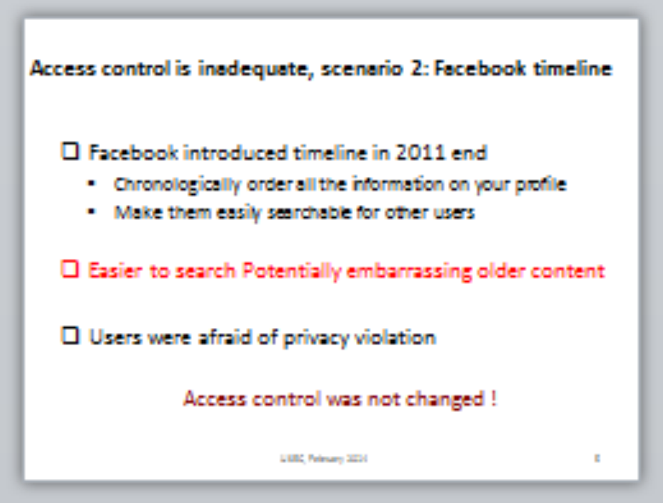
A few words about

Slide Design

No sense of style?

Don’t worry

The most important aspects of slide design have **nothing** to do with style

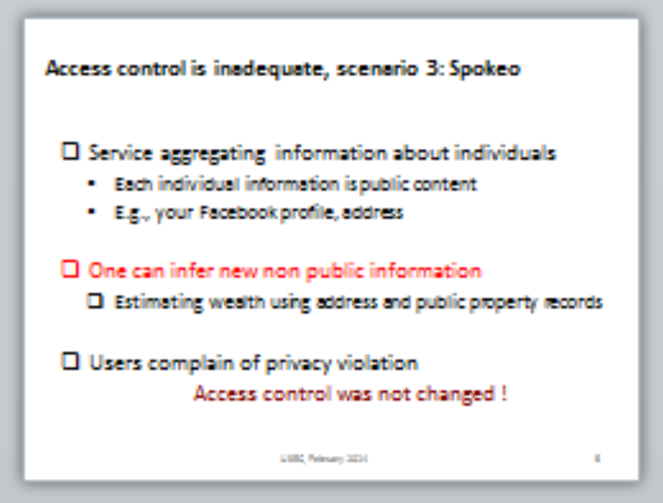


Access control is inadequate, scenario 2: Facebook timeline

- □ Facebook introduced timeline in 2011 end
  - Chronologically order all the information on your profile
  - Make them easily searchable for other users
- □ Easier to search Potentially embarrassing older content
- □ Users were afraid of privacy violation

Access control was not changed !

5

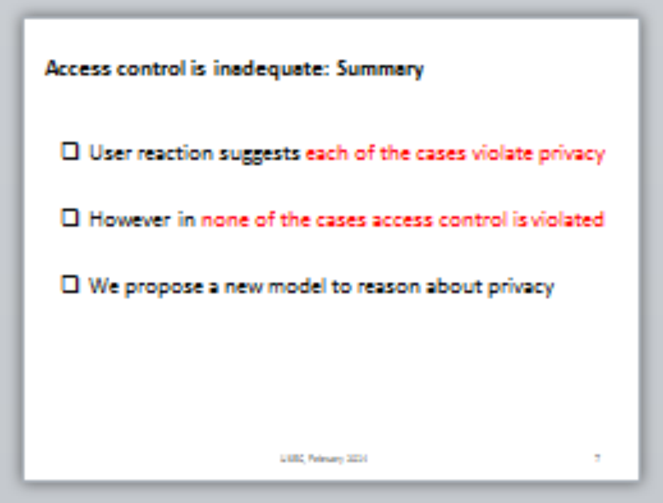


Access control is inadequate, scenario 3: Spokeo

- □ Service aggregating information about individuals
  - Each individual information is public content
  - E.g., your Facebook profile, address
- □ One can infer new non public information
  - □ Estimating wealth using address and public property records
- □ Users complain of privacy violation

Access control was not changed !

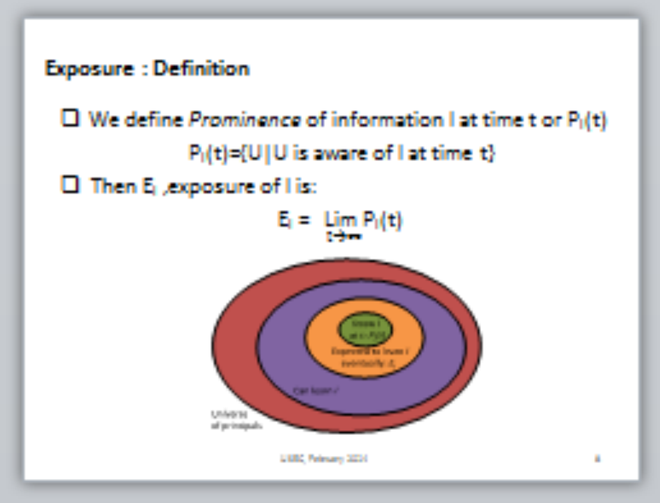
6



Access control is inadequate: Summary

- □ User reaction suggests each of the cases violate privacy
- □ However in none of the cases access control is violated
- □ We propose a new model to reason about privacy

7



Exposure : Definition

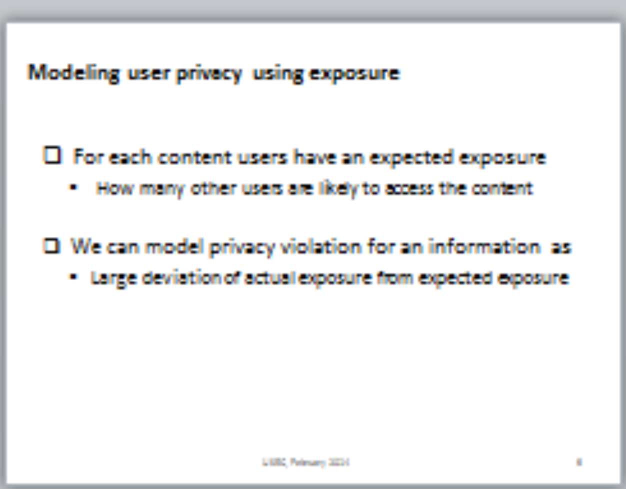
- □ We define *Prominence* of information I at time t or $P_I(t)$

$$P_I(t)=\{U \mid U \text{ is aware of } I \text{ at time } t\}$$

- □ Then E_I ,exposure of I is:

$$E_I = \lim_{t \to \infty} P_I(t)$$

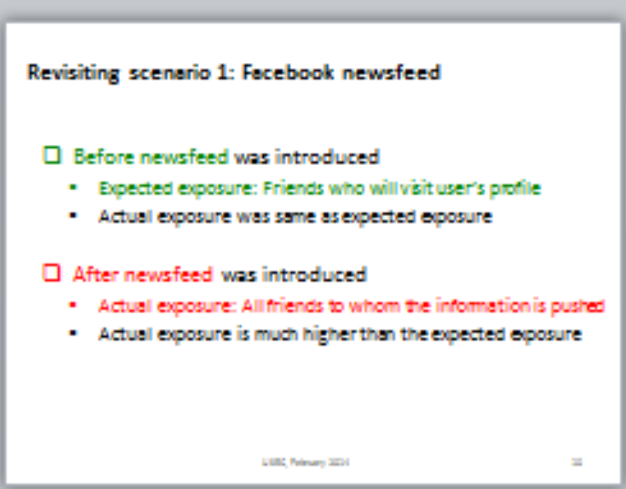
8



Modeling user privacy using exposure

- □ For each content users have an expected exposure
  - How many other users are likely to access the content
- □ We can model privacy violation for an information as
  - Large deviation of actual exposure from expected exposure

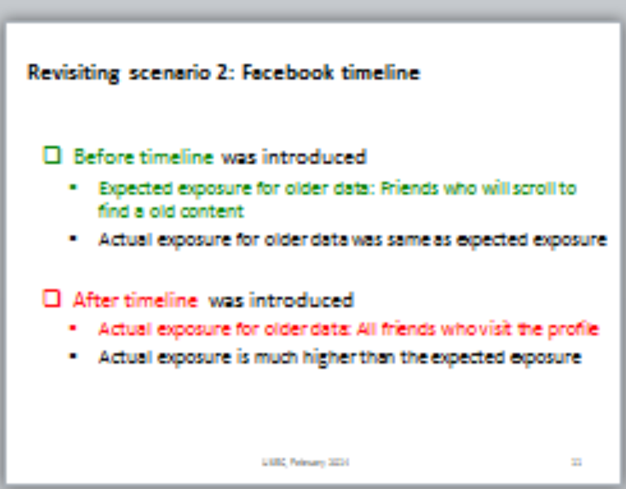
9



Revisiting scenario 1: Facebook newsfeed

- □ Before newsfeed was introduced
  - Expected exposure: Friends who will visit user's profile
  - Actual exposure was same as expected exposure
- □ After newsfeed was introduced
  - Actual exposure: All friends to whom the information is pushed
  - Actual exposure is much higher than the expected exposure

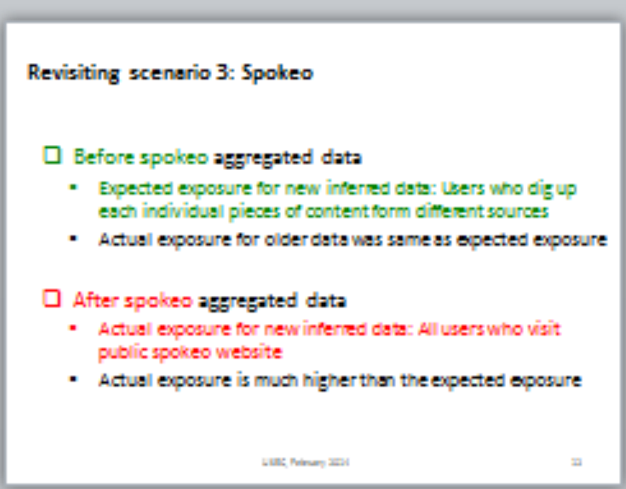
10



Revisiting scenario 2: Facebook timeline

- □ Before timeline was introduced
  - Expected exposure for older data: Friends who will scroll to find a old content
  - Actual exposure for older data was same as expected exposure
- □ After timeline was introduced
  - Actual exposure for older data: All friends who visit the profile
  - Actual exposure is much higher than the expected exposure

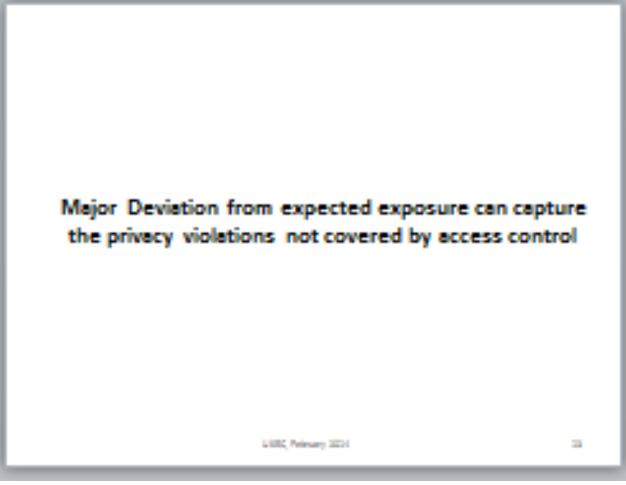
11



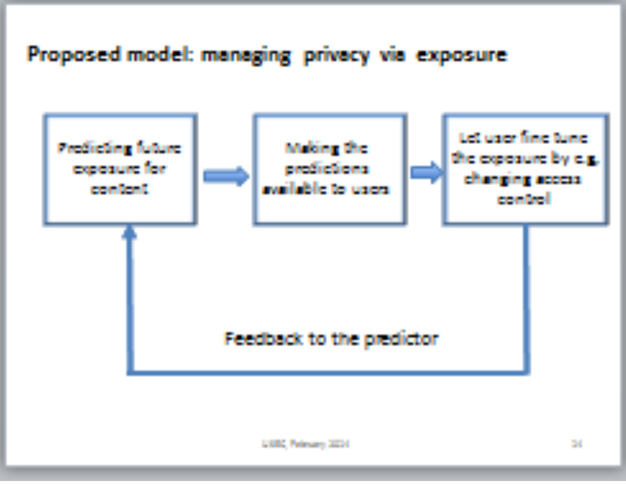
Revisiting scenario 3: Spokeo

- □ Before spokeo aggregated data
  - Expected exposure for new inferred data: Users who dig up each individual pieces of content form different sources
  - Actual exposure for older data was same as expected exposure
- □ After spokeo aggregated data
  - Actual exposure for new inferred data: All users who visit public spokeo website
  - Actual exposure is much higher than the expected exposure

12

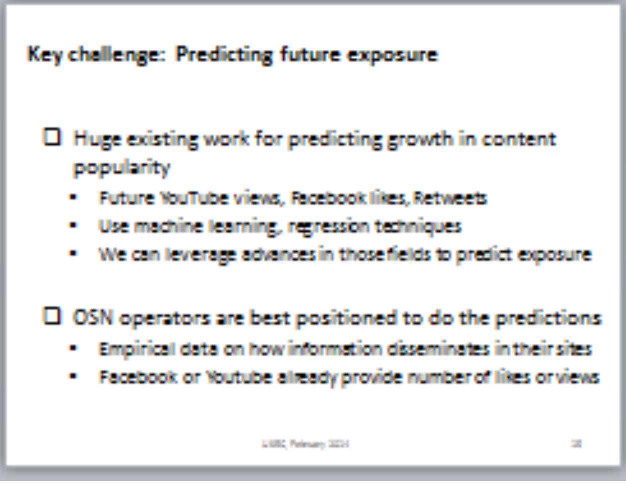


Major Deviation from expected exposure can capture the privacy violations not covered by access control



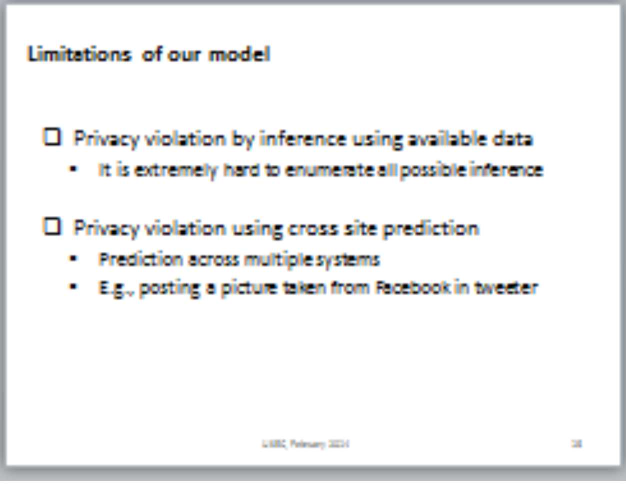
Proposed model: managing privacy via exposure

Predicting future exposure for content
Making the predictions available to users
Let user fine tune the exposure by e.g. changing access control
Feedback to the predictor



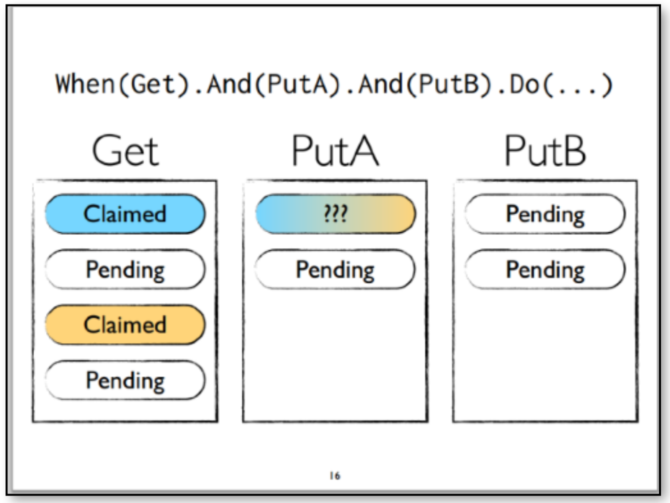
Key challenge: Predicting future exposure

- □ Huge existing work for predicting growth in content popularity
  - Future YouTube views, Facebook likes, Retweets
  - Use machine learning, regression techniques
  - We can leverage advances in those fields to predict exposure
- □ OSN operators are best positioned to do the predictions
  - Empirical data on how information disseminates in their sites
  - Facebook or Youtube already provide number of likes or views

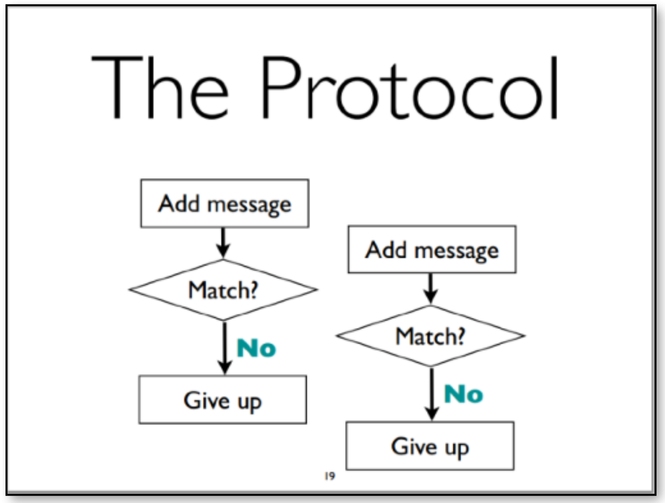


Limitations of our model

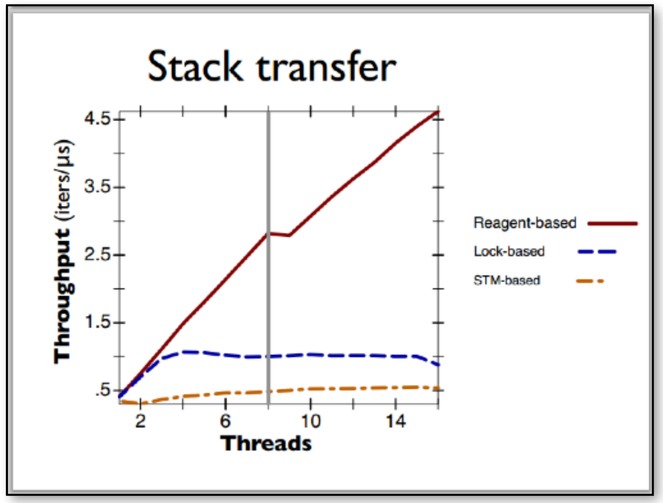
- □ Privacy violation by inference using available data
  - It is extremely hard to enumerate all possible inference
- □ Privacy violation using cross site prediction
  - Prediction across multiple systems
  - E.g., posting a picture taken from Facebook in tweeter



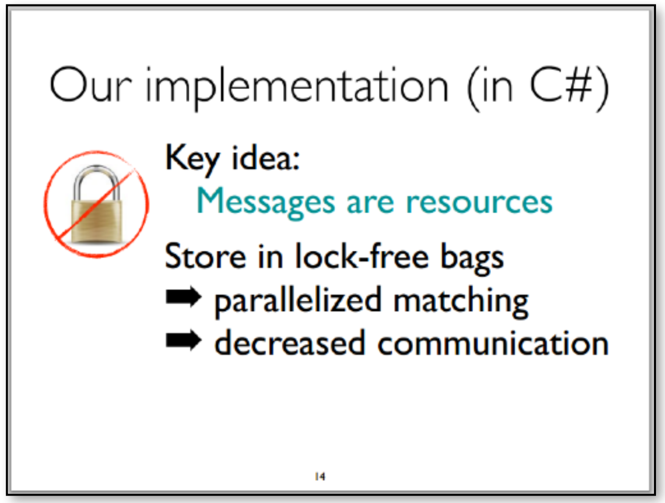
`When(Get).And(PutA).And(PutB).Do(...)`



The Protocol



Stack transfer



Our implementation (in C#)

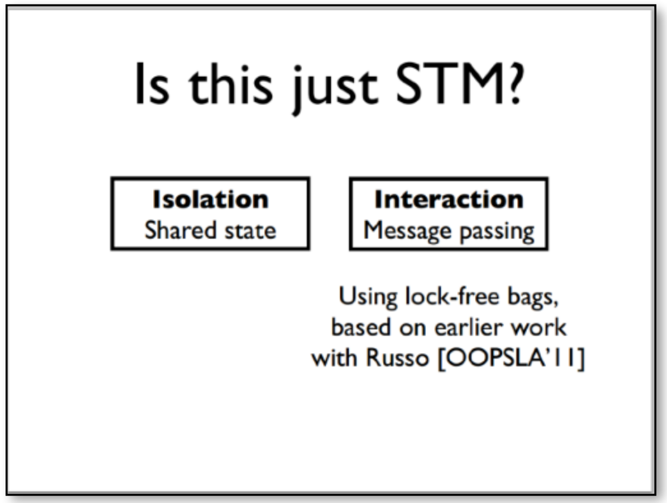
Key idea:

Messages are resources

Store in lock-free bags

➡ parallelized matching

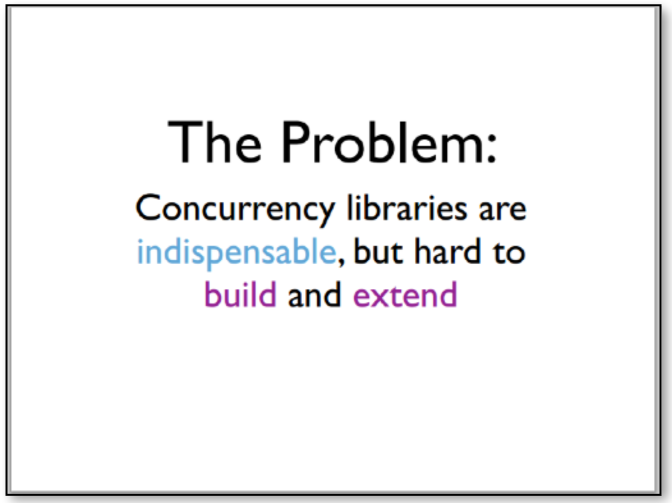
➡ decreased communication



Is this just STM?

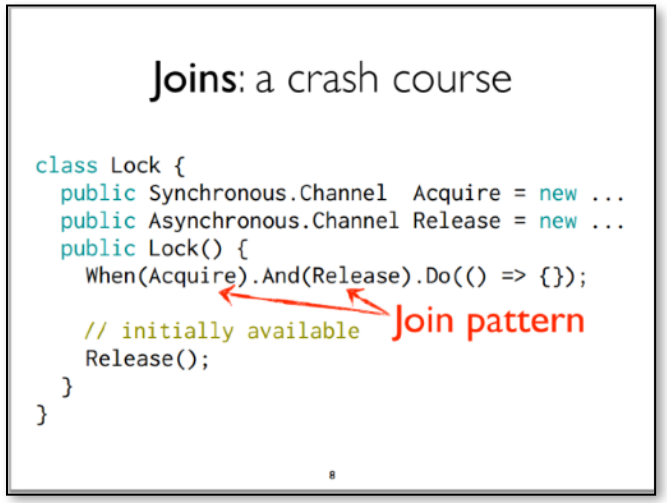
Isolation
Shared state
Interaction
Message passing

Using lock-free bags,
based on earlier work
with Russo [OOPSLA'11]



The Problem:

Concurrency libraries are indispensable, but hard to build and extend

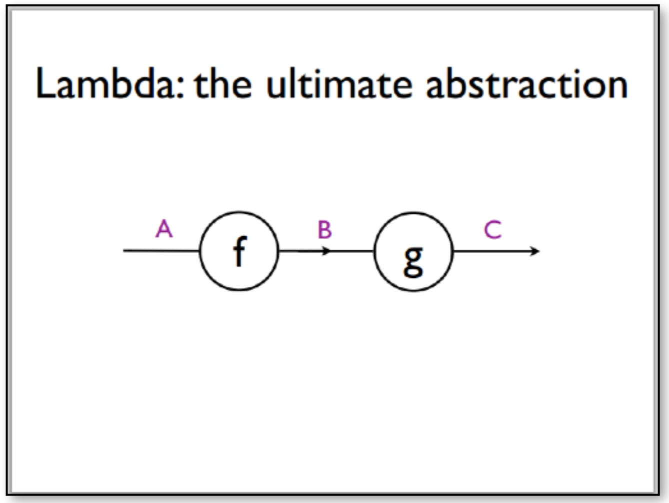


Joins: a crash course

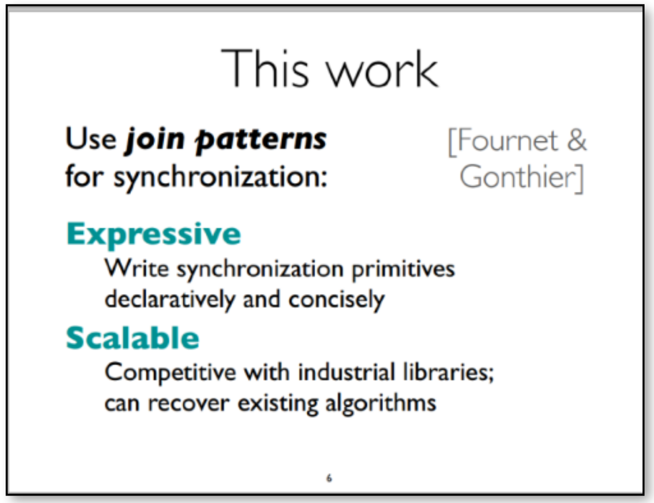
```
class Lock {
  public Synchronous.Channel  Acquire = new ...
  public Asynchronous.Channel Release = new ...
  public Lock() {
    When(Acquire).And(Release).Do(() => {});

    // initially available
    Release();
  }
}
```

Join pattern



Lambda: the ultimate abstraction



This work

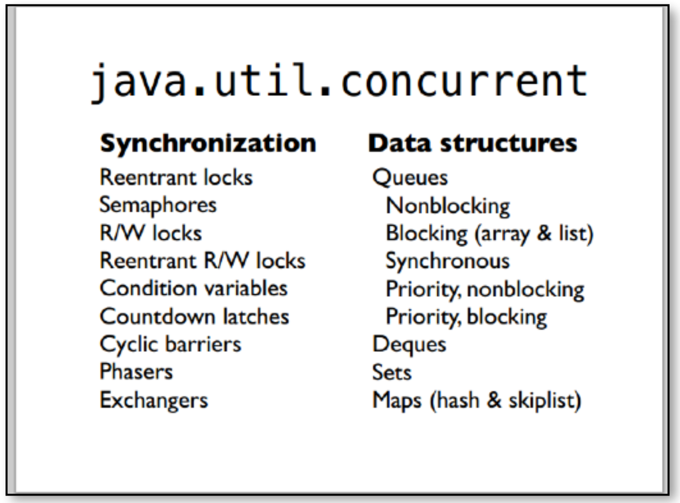
Use ***join patterns*** for synchronization: [Fournet & Gonthier]

Expressive

Write synchronization primitives declaratively and concisely

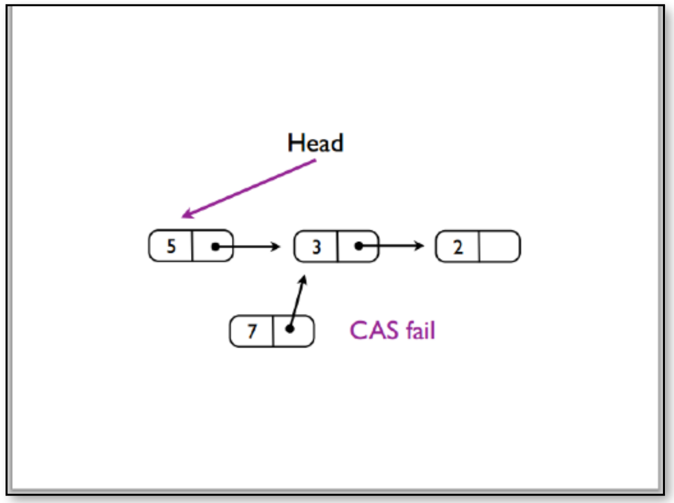
Scalable

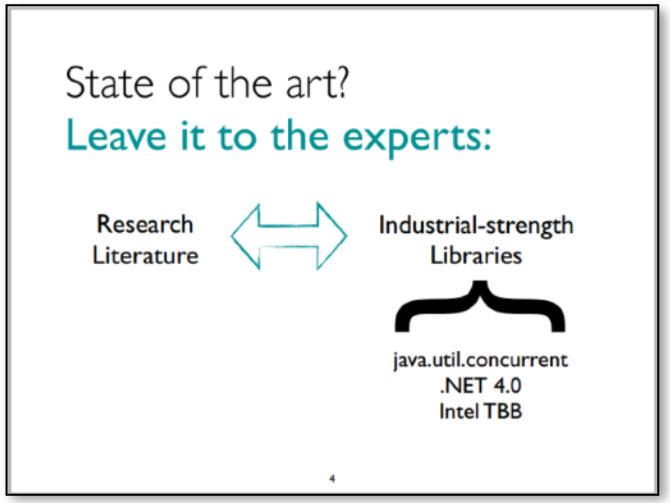
Competitive with industrial libraries; can recover existing algorithms



`java.util.concurrent`

Synchronization	**Data structures**
Reentrant locks	Queues
Semaphores	Nonblocking
R/W locks	Blocking (array & list)
Reentrant R/W locks	Synchronous
Condition variables	Priority, nonblocking
Countdown latches	Priority, blocking
Cyclic barriers	Deques
Phasers	Sets
Exchangers	Maps (hash & skiplist)





State of the art?
Leave it to the experts:

java.util.concurrent
.NET 4.0
Intel TBB

Key takeaways

- **Avoid PowerPoint-itis**
  - Don't put lots of text on slides just so they are readable independently of the talk
- **Vary the look of the slides**
  - Some text-only slides are fine, but if there are too many in a row, audience falls asleep

That's all Folks!