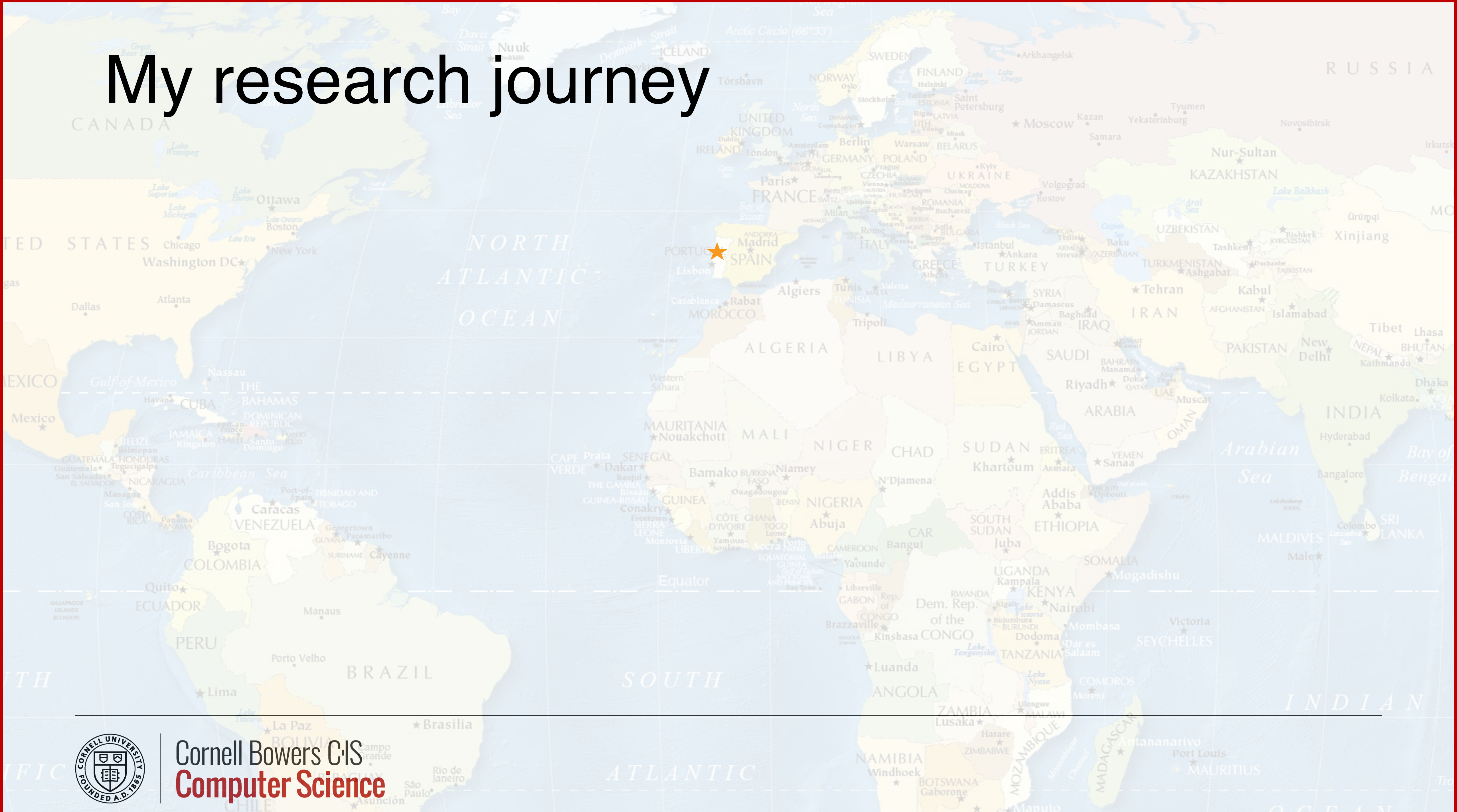
# Finding a research topic

## or being found by a research topic?

Alexandra Silva

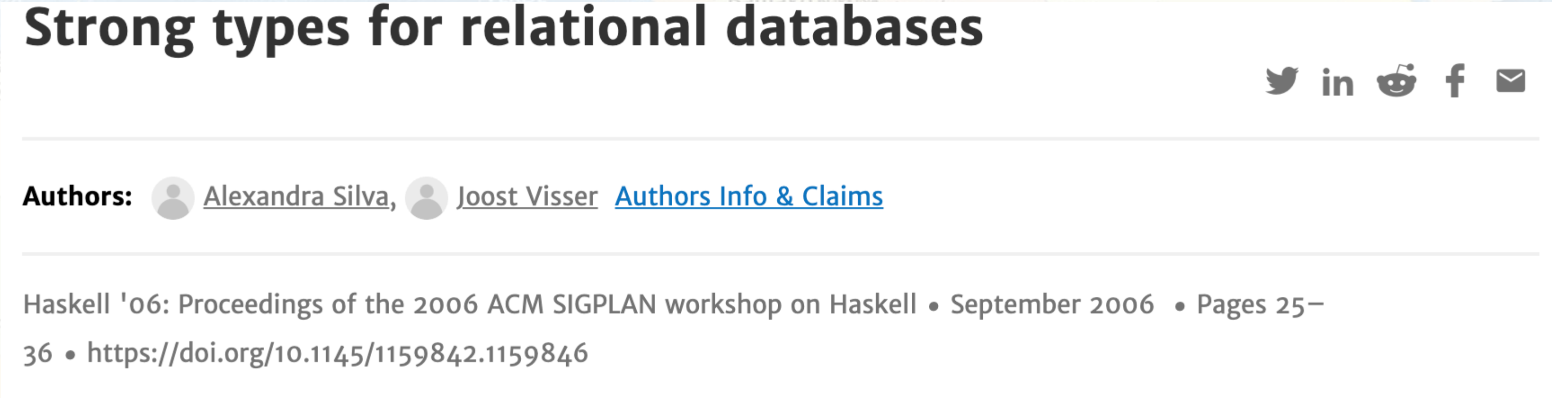
Cornell Bowers CIS
Computer Science

# My research journey



# My research journey

Functional Programming

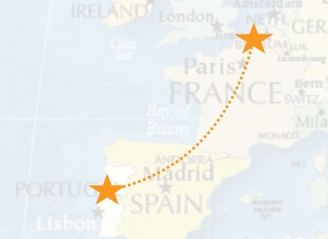


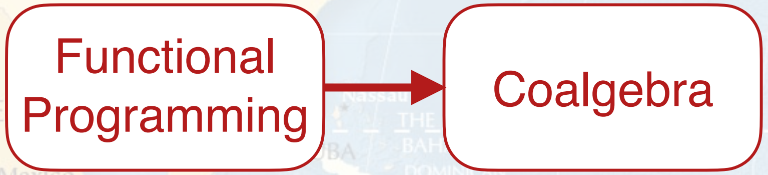
## Strong types for relational databases

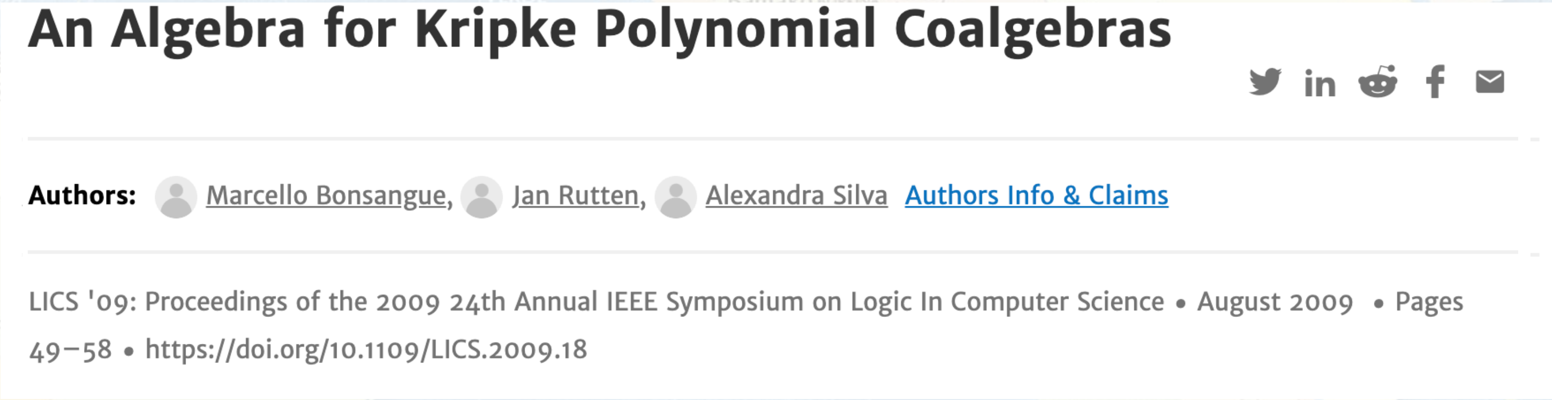
**Authors:** Alexandra Silva, Joost Visser Authors Info & Claims

Haskell '06: Proceedings of the 2006 ACM SIGPLAN workshop on Haskell • September 2006 • Pages 25–36 • https://doi.org/10.1145/1159842.1159846

# My research journey




Functional Programming
Coalgebra

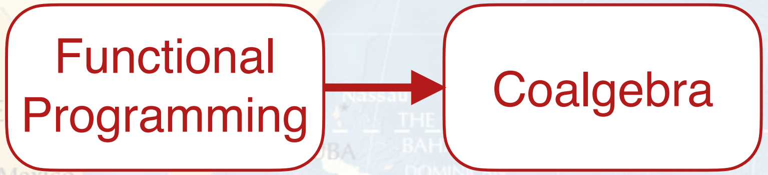


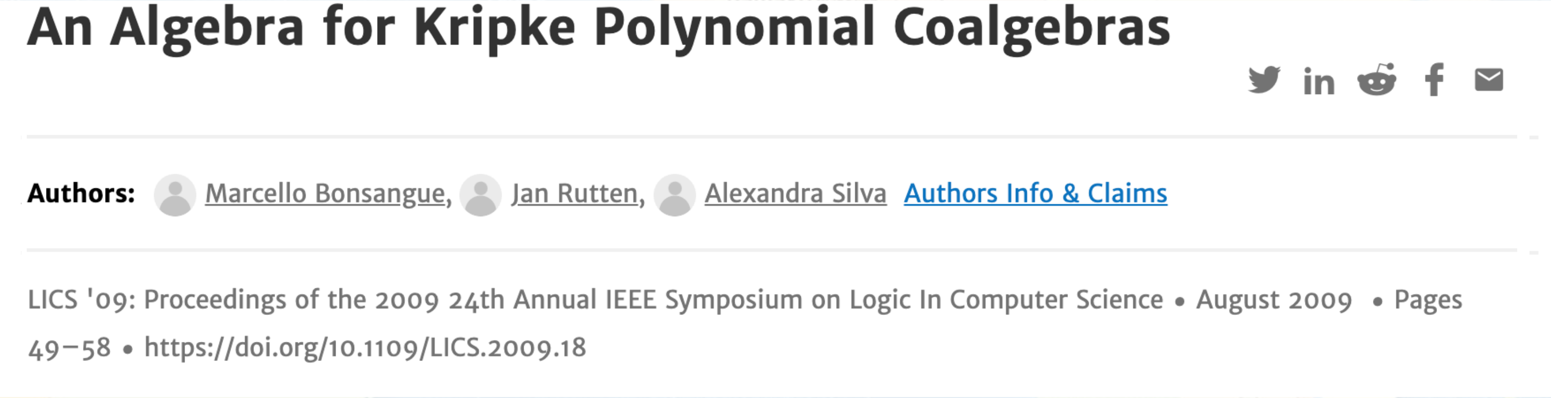
## An Algebra for Kripke Polynomial Coalgebras

**Authors:** Marcello Bonsangue, Jan Rutten, Alexandra Silva Authors Info & Claims

LICS '09: Proceedings of the 2009 24th Annual IEEE Symposium on Logic In Computer Science • August 2009 • Pages 49–58 • https://doi.org/10.1109/LICS.2009.18

# My research journey



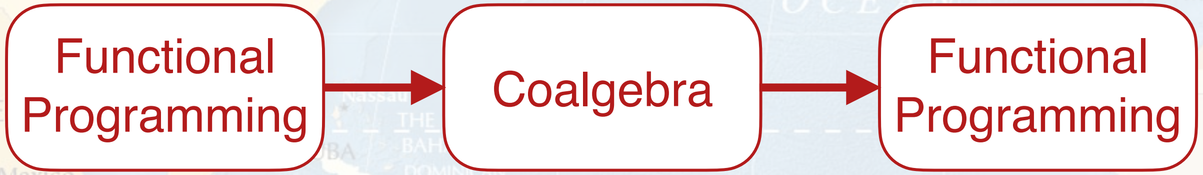


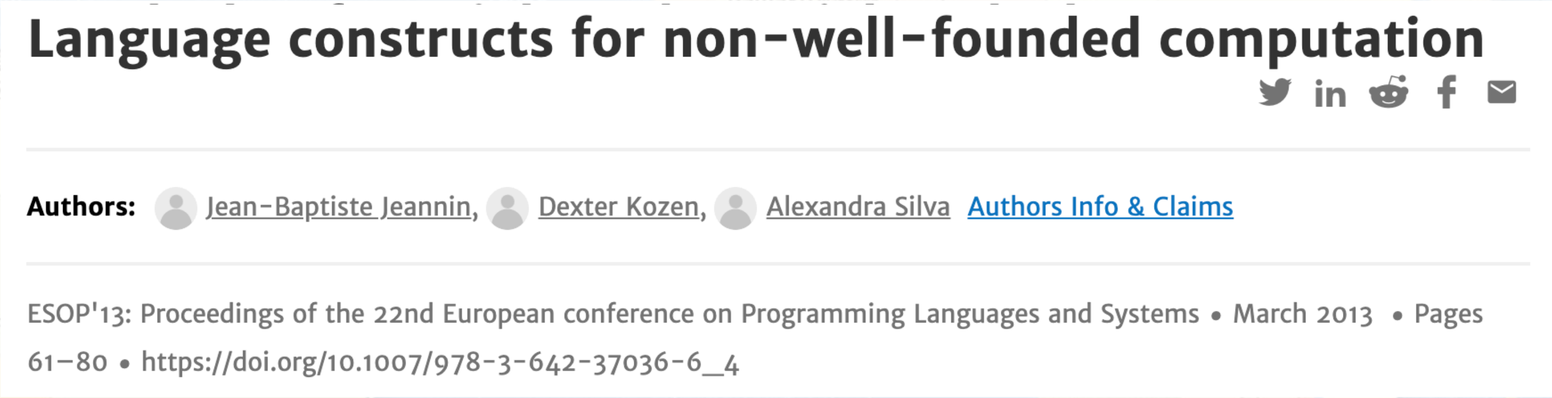
## An Algebra for Kripke Polynomial Coalgebras

**Authors:** Marcello Bonsangue, Jan Rutten, Alexandra Silva Authors Info & Claims

LICS '09: Proceedings of the 2009 24th Annual IEEE Symposium on Logic In Computer Science • August 2009 • Pages 49–58 • https://doi.org/10.1109/LICS.2009.18

# My research journey

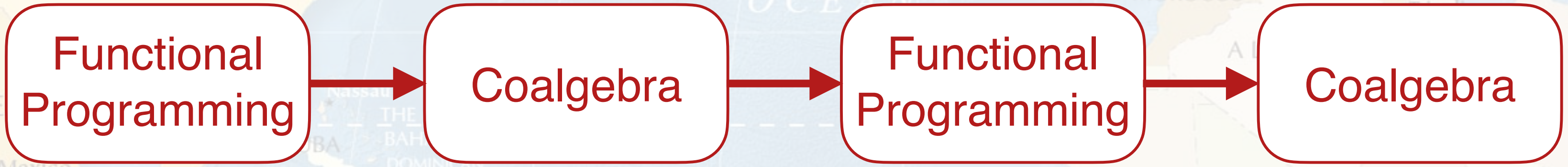

Functional Programming → Coalgebra → Functional Programming


**Language constructs for non-well-founded computation**

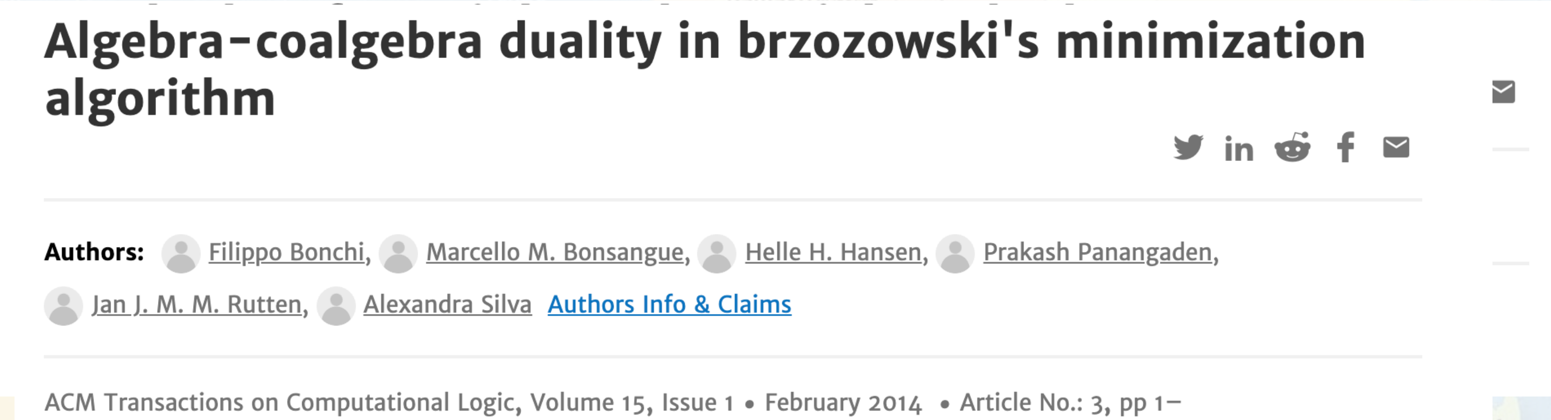
**Authors:** Jean-Baptiste Jeannin, Dexter Kozen, Alexandra Silva Authors Info & Claims

ESOP'13: Proceedings of the 22nd European conference on Programming Languages and Systems • March 2013 • Pages 61–80 • https://doi.org/10.1007/978-3-642-37036-6_4

# My research journey



Functional Programming → Coalgebra → Functional Programming → Coalgebra



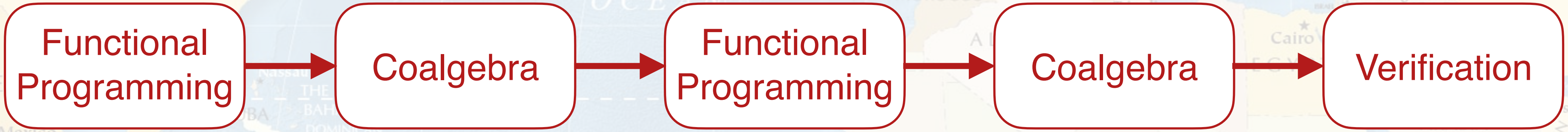
**Algebra-coalgebra duality in brzozowski's minimization algorithm**

**Authors:** Filippo Bonchi, Marcello M. Bonsangue, Helle H. Hansen, Prakash Panangaden, Jan J. M. M. Rutten, Alexandra Silva Authors Info & Claims

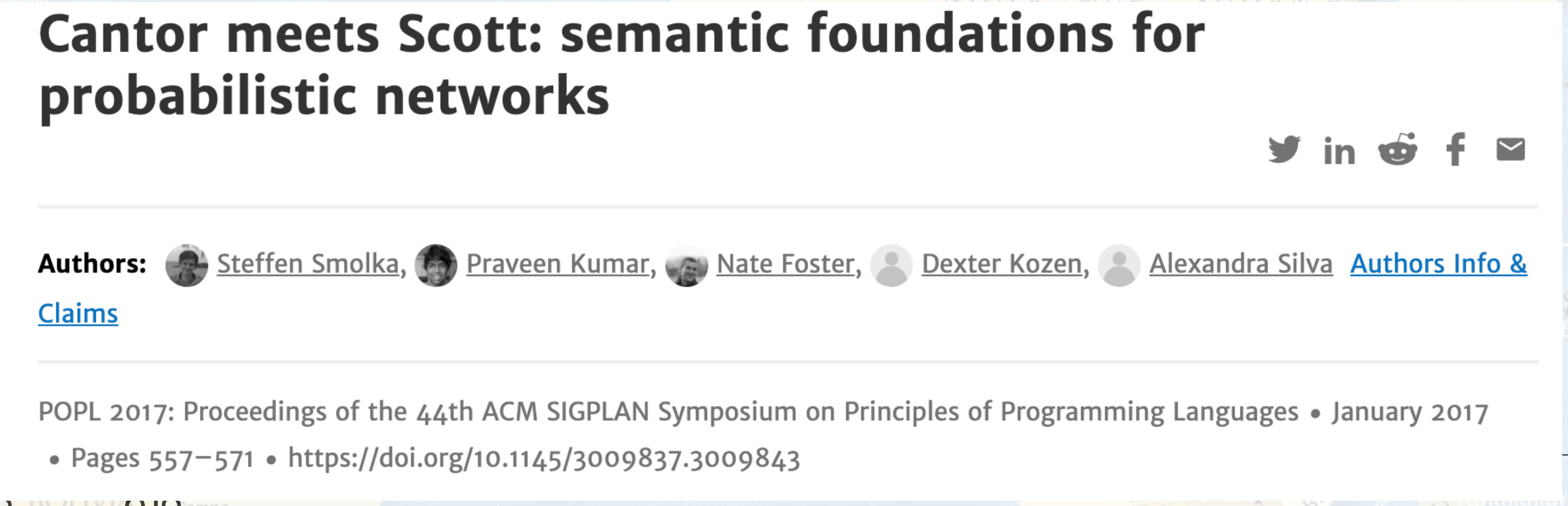
ACM Transactions on Computational Logic, Volume 15, Issue 1 • February 2014 • Article No.: 3, pp 1–

# My research journey

Ottawa
Boston
Chicago
Washington DC
NORTH
ATLANTIC
FRANCE
SPAIN
Madrid
Lisbon
GERMANY



Functional Programming → Coalgebra → Functional Programming → Coalgebra → Verification

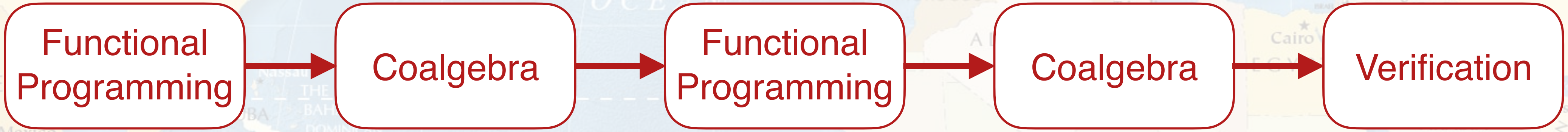


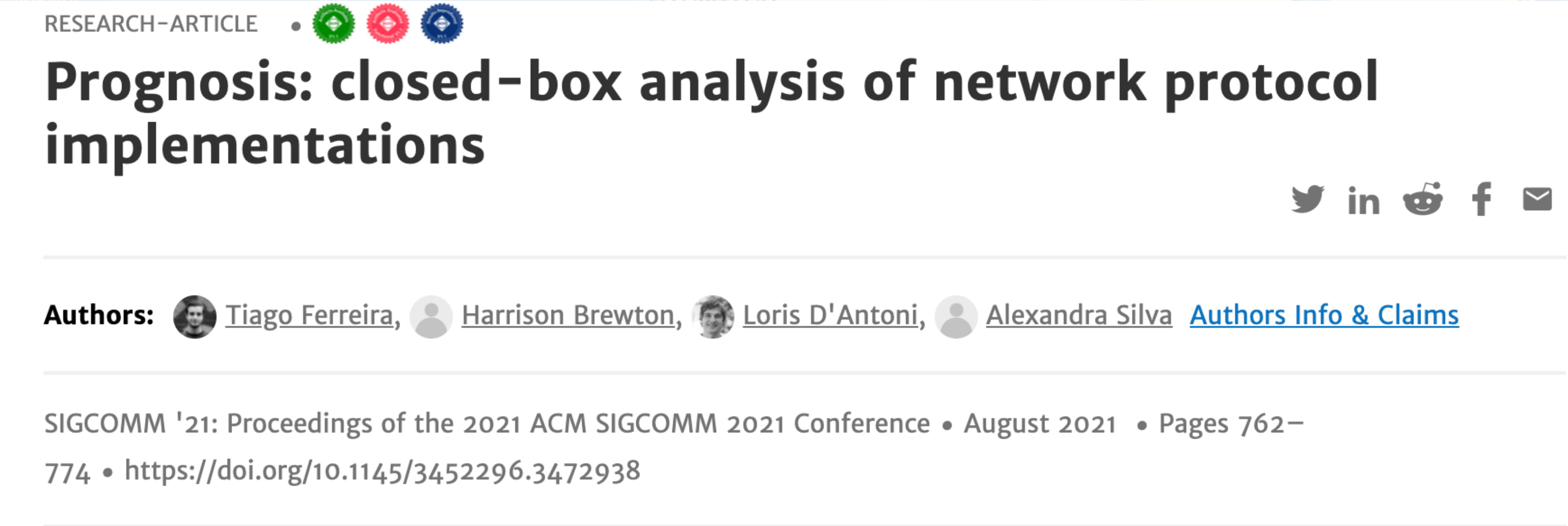
**Cantor meets Scott: semantic foundations for probabilistic networks**

**Authors:** Steffen Smolka, Praveen Kumar, Nate Foster, Dexter Kozen, Alexandra Silva Authors Info & Claims

POPL 2017: Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages • January 2017 • Pages 557–571 • https://doi.org/10.1145/3009837.3009843

# My research journey


Functional Programming → Coalgebra → Functional Programming → Coalgebra → Verification

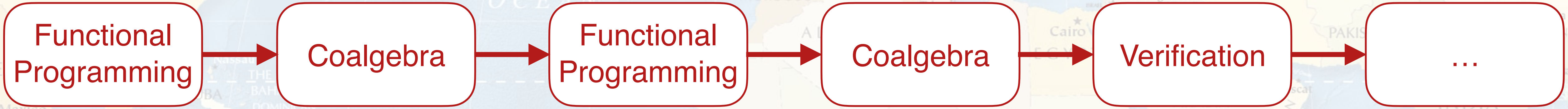

RESEARCH-ARTICLE •

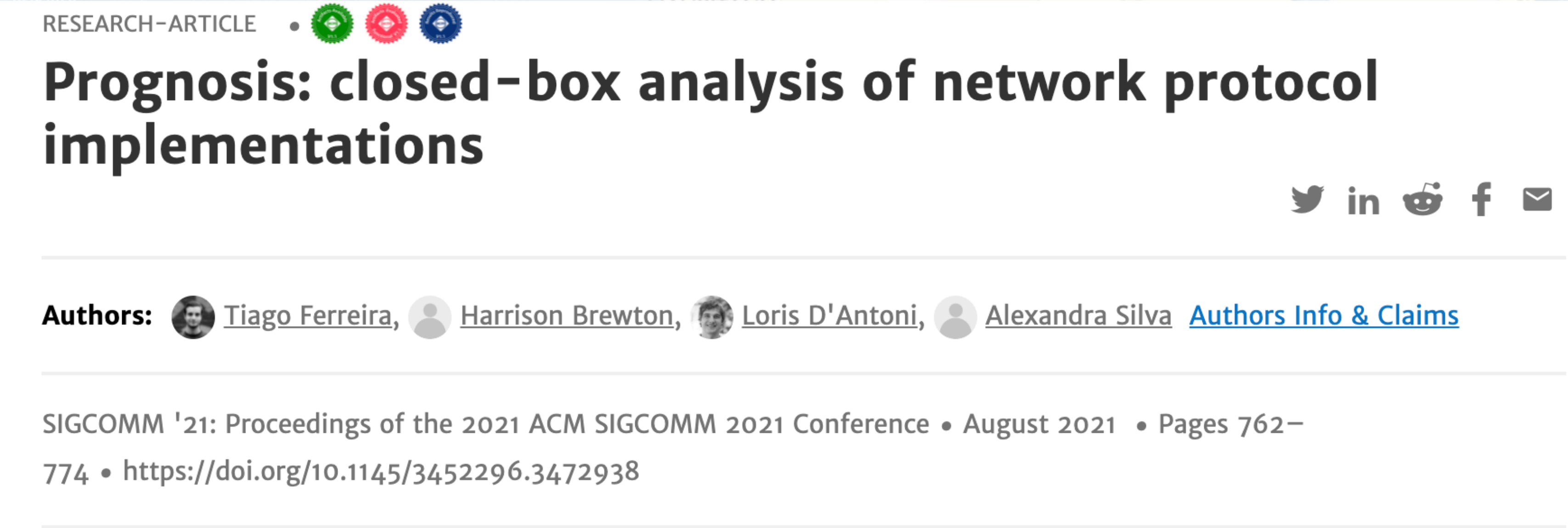
## Prognosis: closed-box analysis of network protocol implementations

**Authors:** Tiago Ferreira, Harrison Brewton, Loris D'Antoni, Alexandra Silva Authors Info & Claims

SIGCOMM '21: Proceedings of the 2021 ACM SIGCOMM 2021 Conference • August 2021 • Pages 762–774 • https://doi.org/10.1145/3452296.3472938

# My research journey


Functional Programming → Coalgebra → Functional Programming → Coalgebra → Verification → …

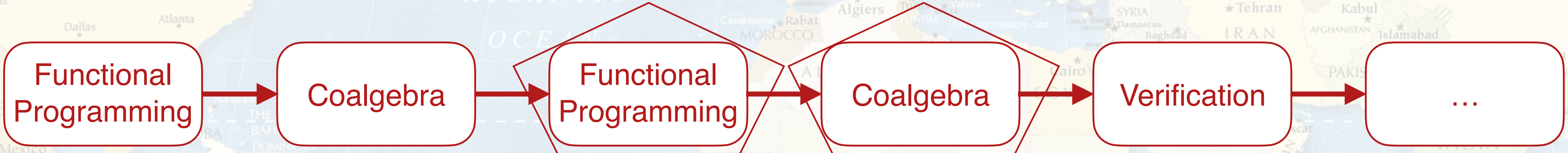

RESEARCH-ARTICLE •

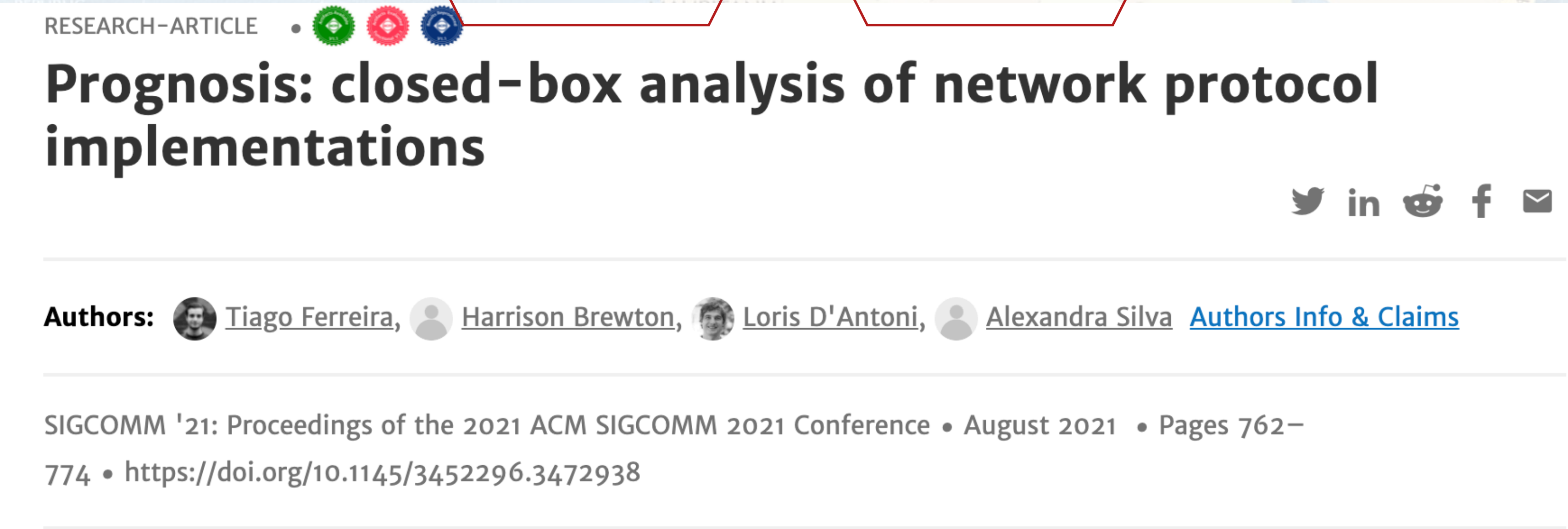
## Prognosis: closed-box analysis of network protocol implementations

**Authors:** Tiago Ferreira, Harrison Brewton, Loris D'Antoni, Alexandra Silva Authors Info & Claims

SIGCOMM '21: Proceedings of the 2021 ACM SIGCOMM 2021 Conference • August 2021 • Pages 762–774 • https://doi.org/10.1145/3452296.3472938

# My research journey





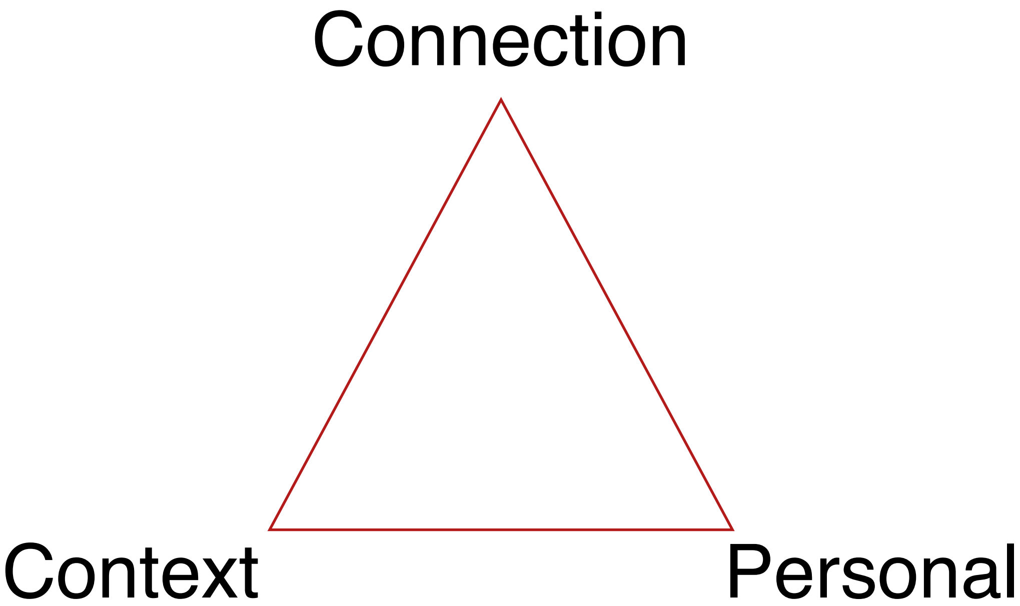
RESEARCH-ARTICLE •

**Prognosis: closed-box analysis of network protocol implementations**

Authors: Tiago Ferreira, Harrison Brewton, Loris D'Antoni, Alexandra Silva Authors Info & Claims

SIGCOMM '21: Proceedings of the 2021 ACM SIGCOMM 2021 Conference • August 2021 • Pages 762–774 • https://doi.org/10.1145/3452296.3472938

# How did I choose a research topic?

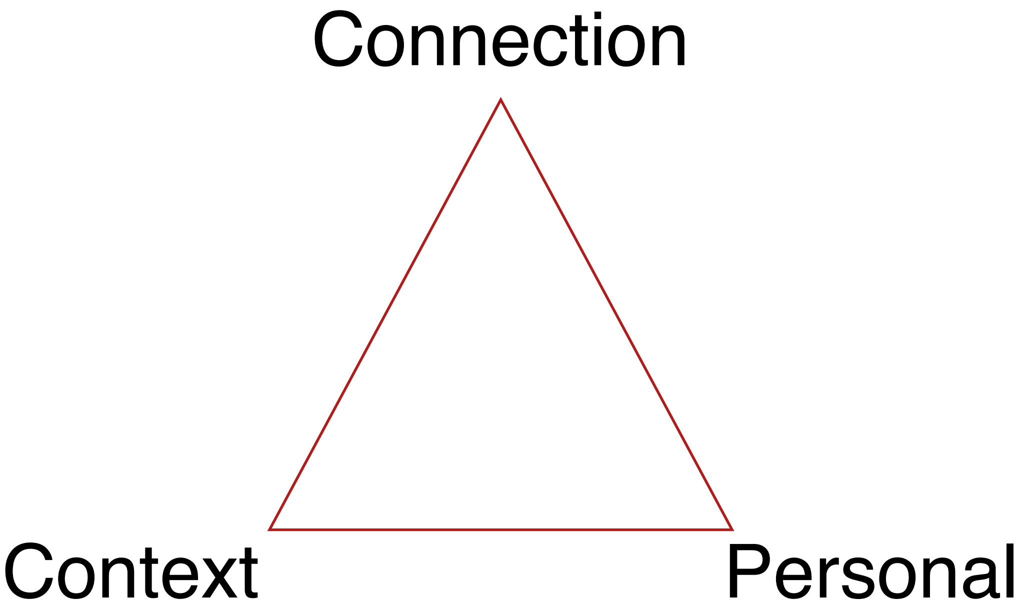


Connection

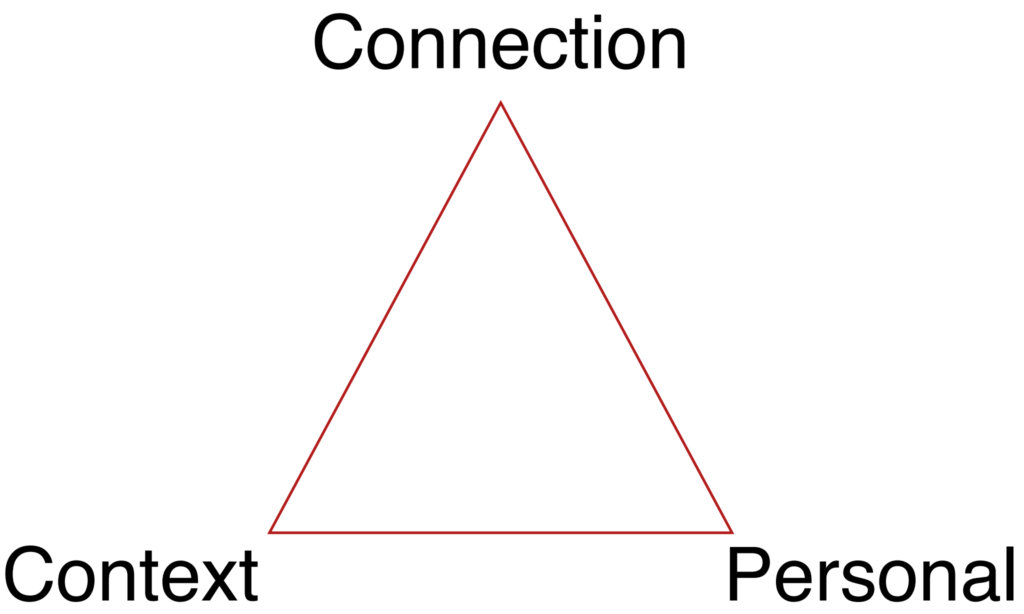
Context

Personal

# How did I choose a research topic?



# How did I choose a research topic?



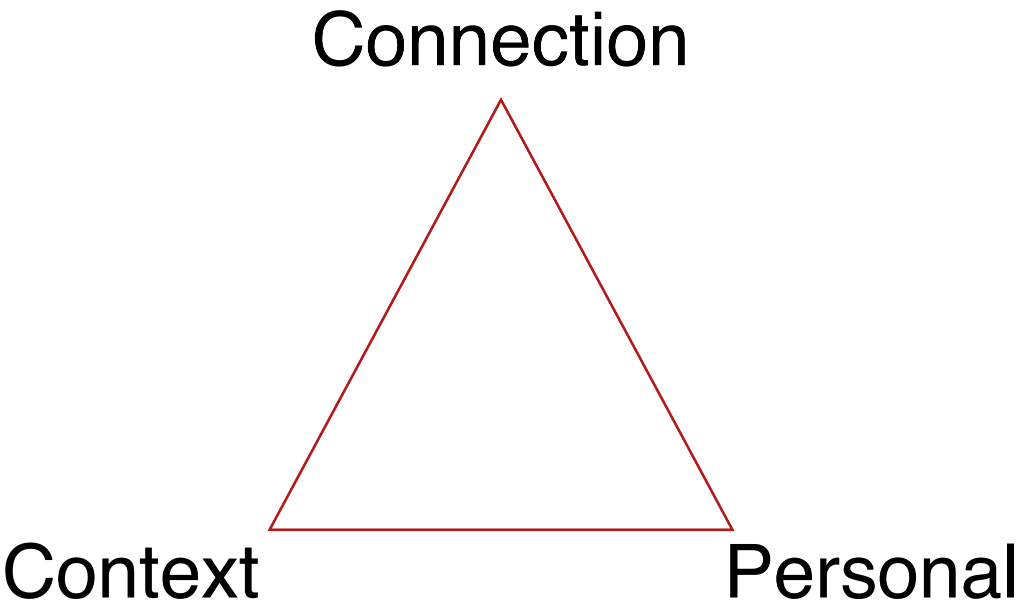
Connection

Context

Personal

**Some lessons I learned along the way:**

# How did I choose a research topic?



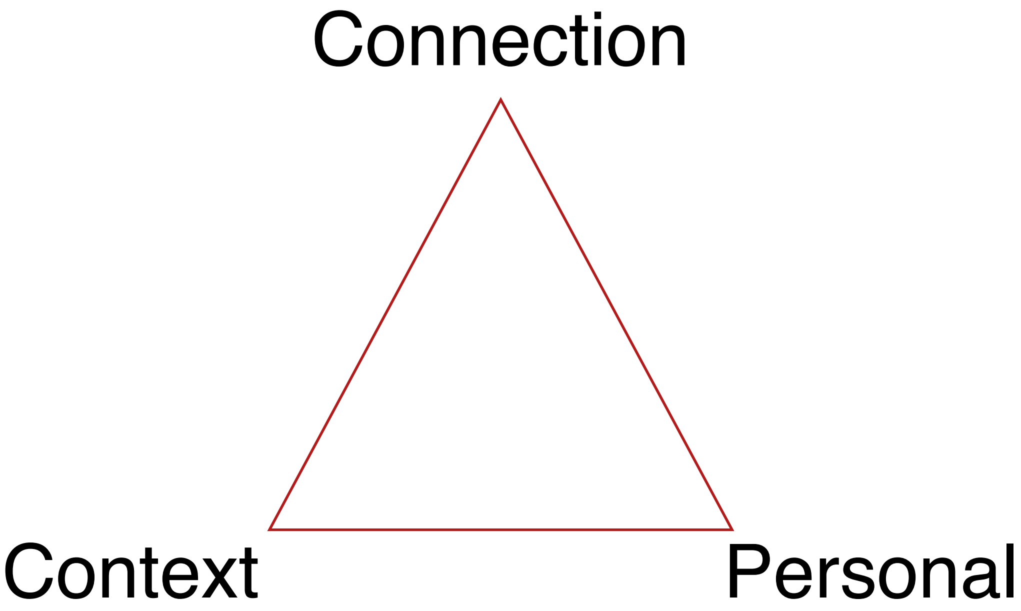
Connection

Context

Personal

**Some lessons I learned along the way:**

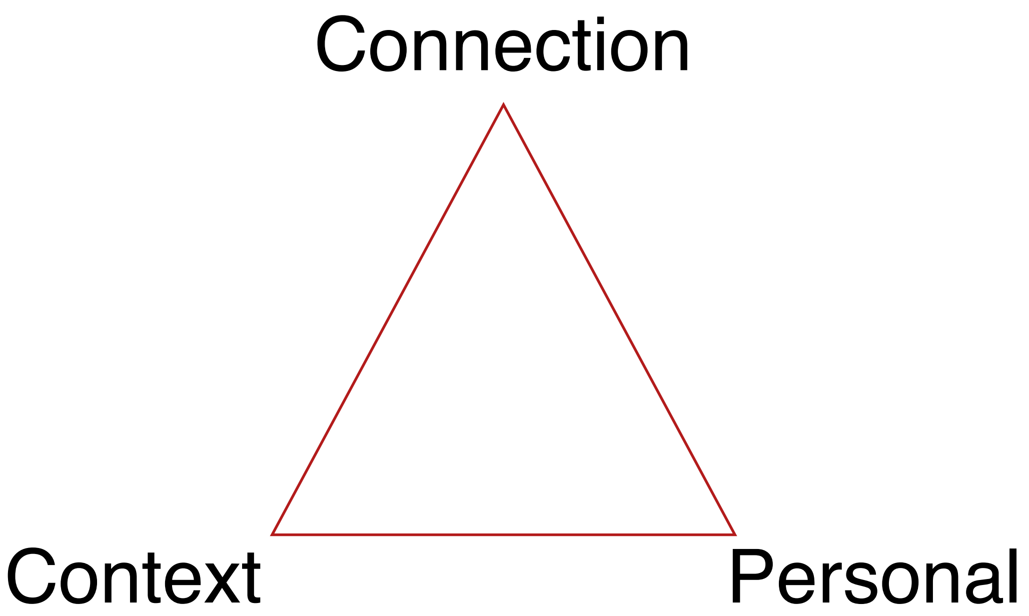
# How did I choose a research topic?



**Some lessons I learned along the way:**

- The topic is only one component

# How did I choose a research topic?



Connection

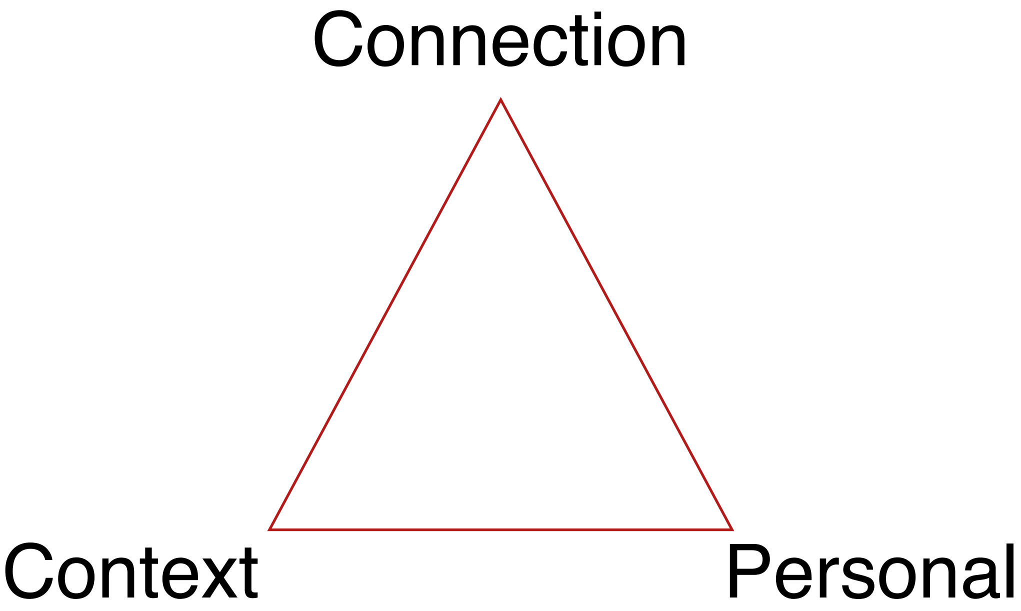
Context

Personal

**Some lessons I learned along the way:**

- The topic is only one component
- Working with other people implies compromise

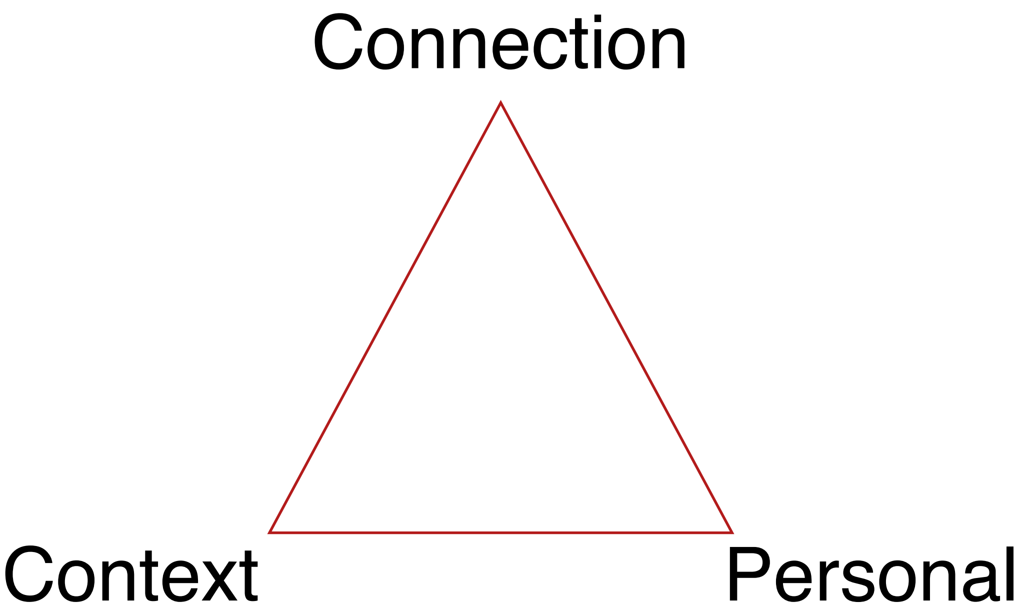
# How did I choose a research topic?



**Some lessons I learned along the way:**

- The topic is only one component
- Working with other people implies compromise
- There are things I will not work on

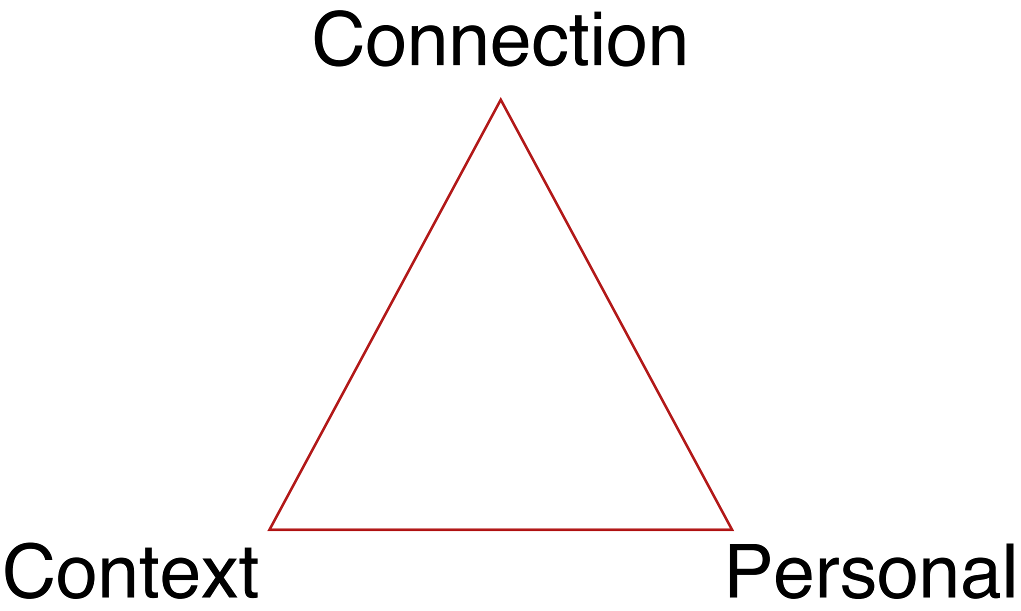
# How did I choose a research topic?



**Some lessons I learned along the way:**

- The topic is only one component
- Working with other people implies compromise
- There are things I will not work on
- There are people I will not work with

# How did I choose a research topic?



**Some lessons I learned along the way:**

- The topic is only one component
- Working with other people implies compromise
- There are things I will not work on
- There are people I will not work with
- What you do not want is as important as what you want

# Supervisors (-or- the early days)

# Supervisors (-or- the early days)

# Supervisors (-or- the early days)

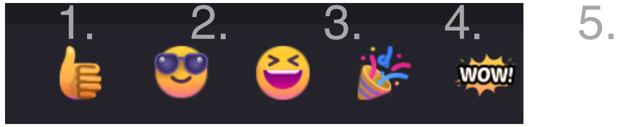
Which of these is most important in a supervisor:

1. Experience
2. Knowledge (breadth, depth)
3. Network
4. Empathy
5. Fun

# Supervisors (-or- the early days)

Which of these is most important in a supervisor:

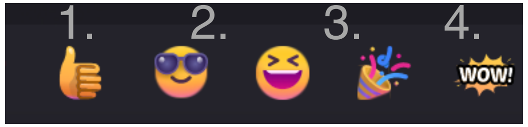
1. Experience
2. Knowledge (breadth, depth)
3. Network
4. Empathy
5. Fun



# Supervisors (-or- the early days)

Which of these is most important in a supervisor:

1. Experience
2. Knowledge (breadth, depth)
3. Network
4. Empathy
5. Fun



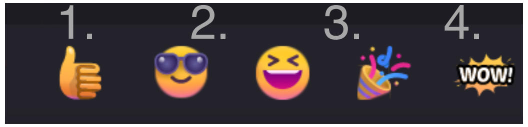
5.

**Others?**

# Supervisors (-or- the early days)

Which of these is most important in a supervisor:

1. Experience
2. Knowledge (breadth, depth)
3. Network
4. Empathy
5. Fun



5.

**Others?**

# Collaborators

# Collaborators

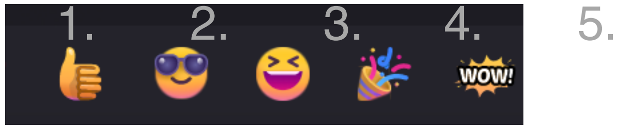
Which of these is most important in a collaborato

1. Experience
2. Knowledge (breadth, depth)
3. Network
4. Empathy
5. Fun

# Collaborators

Which of these is most important in a collaborato

1. Experience
2. Knowledge (breadth, depth)
3. Network
4. Empathy
5. Fun

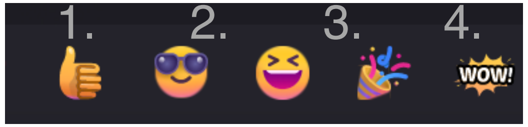


# Collaborators

Which of these is most important in a collaborato

1. Experience
2. Knowledge (breadth, depth)
3. Network
4. Empathy
5. Fun

**Others?**



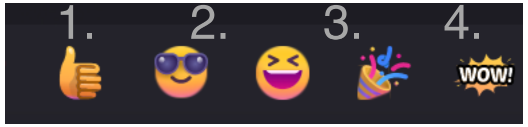
5.

# Collaborators

Which of these is most important in a collaborato

1. Experience
2. Knowledge (breadth, depth)
3. Network
4. Empathy
5. Fun

**Others?**



5.

# Students

# Students

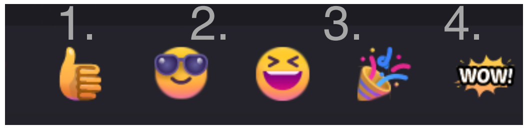
Which of these is most important in a student:

1. Experience
2. Knowledge (breadth, depth)
3. Network
4. Empathy
5. Fun

# Students

Which of these is most important in a student:

1. Experience
2. Knowledge (breadth, depth)
3. Network
4. Empathy
5. Fun



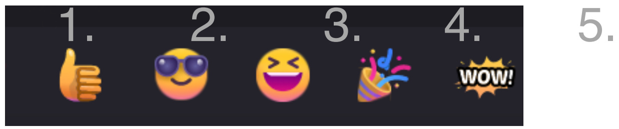
5.

# Students

Which of these is most important in a student:

1. Experience
2. Knowledge (breadth, depth)
3. Network
4. Empathy
5. Fun

**Others?**

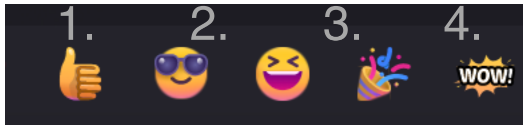


# Students

Which of these is most important in a student:

1. Experience
2. Knowledge (breadth, depth)
3. Network
4. Empathy
5. Fun

**Others?**



5.

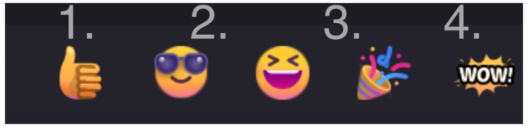
# Community

dreamstime

# Community

Which of these is most important in a research community:

1. Hot topics to work on
2. Mentoring events
3. Collaborative opportunities
4. Good teaching materials
5. Conferences/Networking events



5.

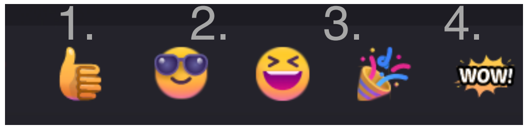
# Community

dreamstime

Which of these is most important in a research community:

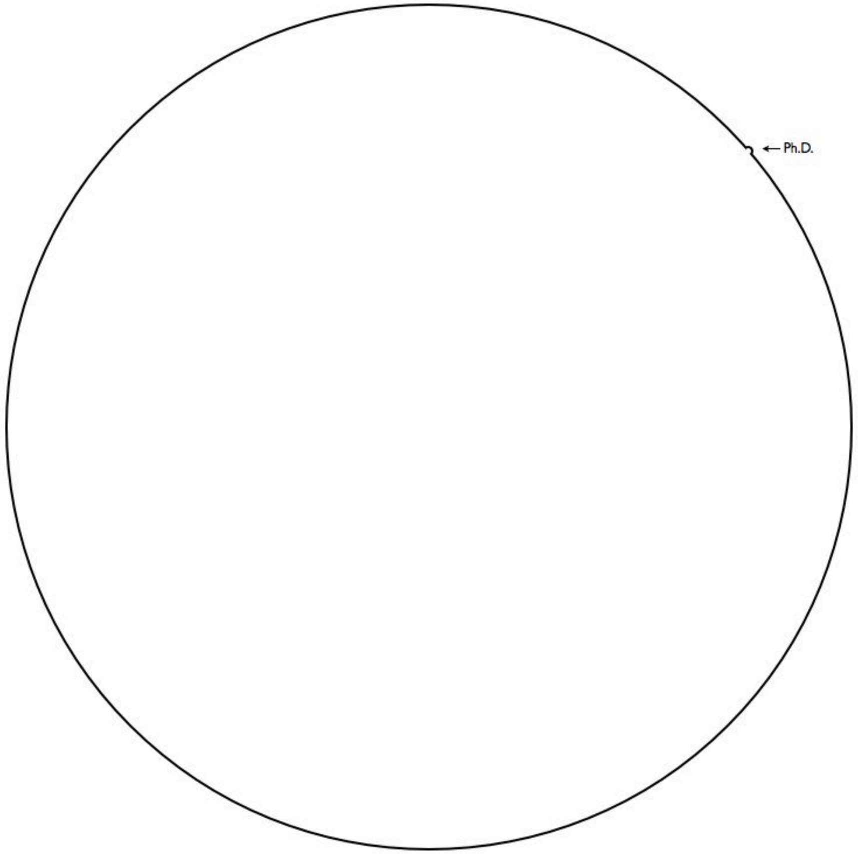
1. Hot topics to work on
2. Mentoring events
3. Collaborative opportunities
4. Good teaching materials
5. Conferences/Networking events

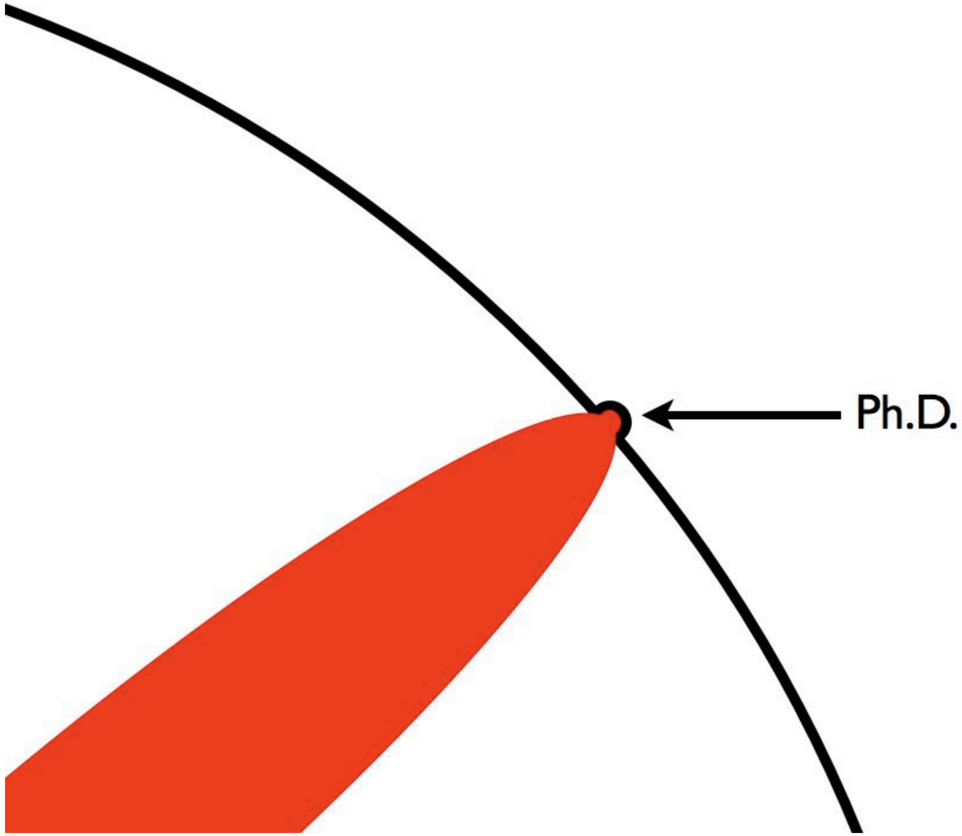
**Others?**



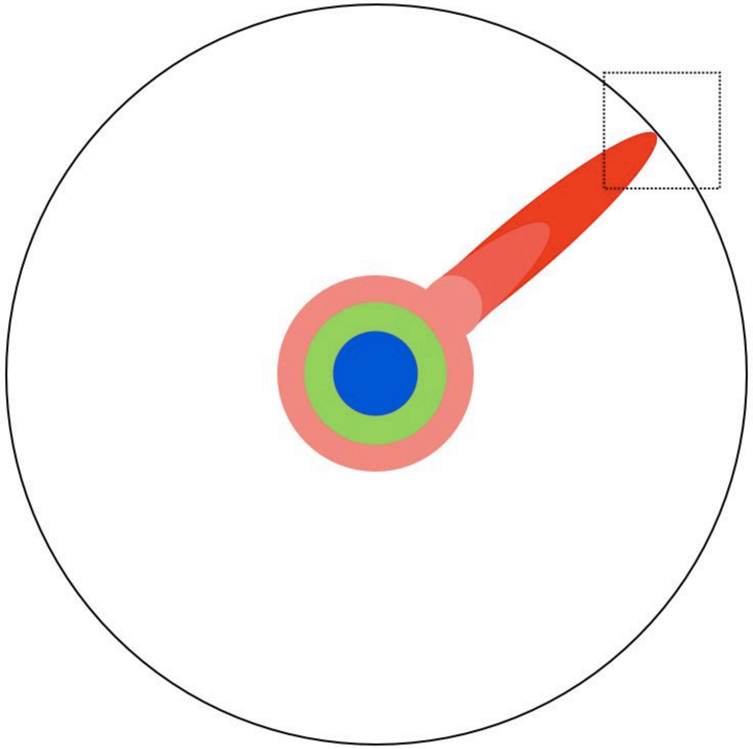
5.

# Enthusiasm matters…

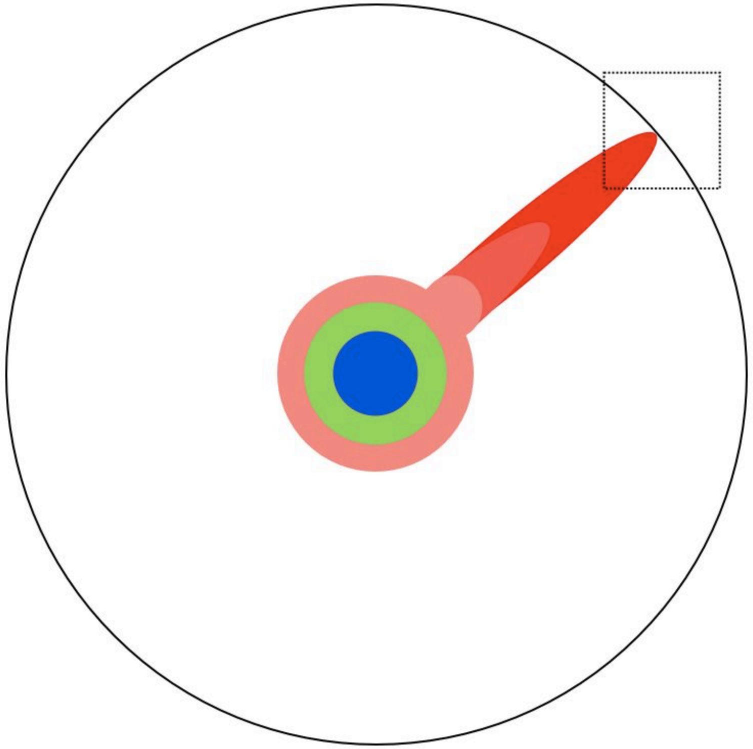




# … but …



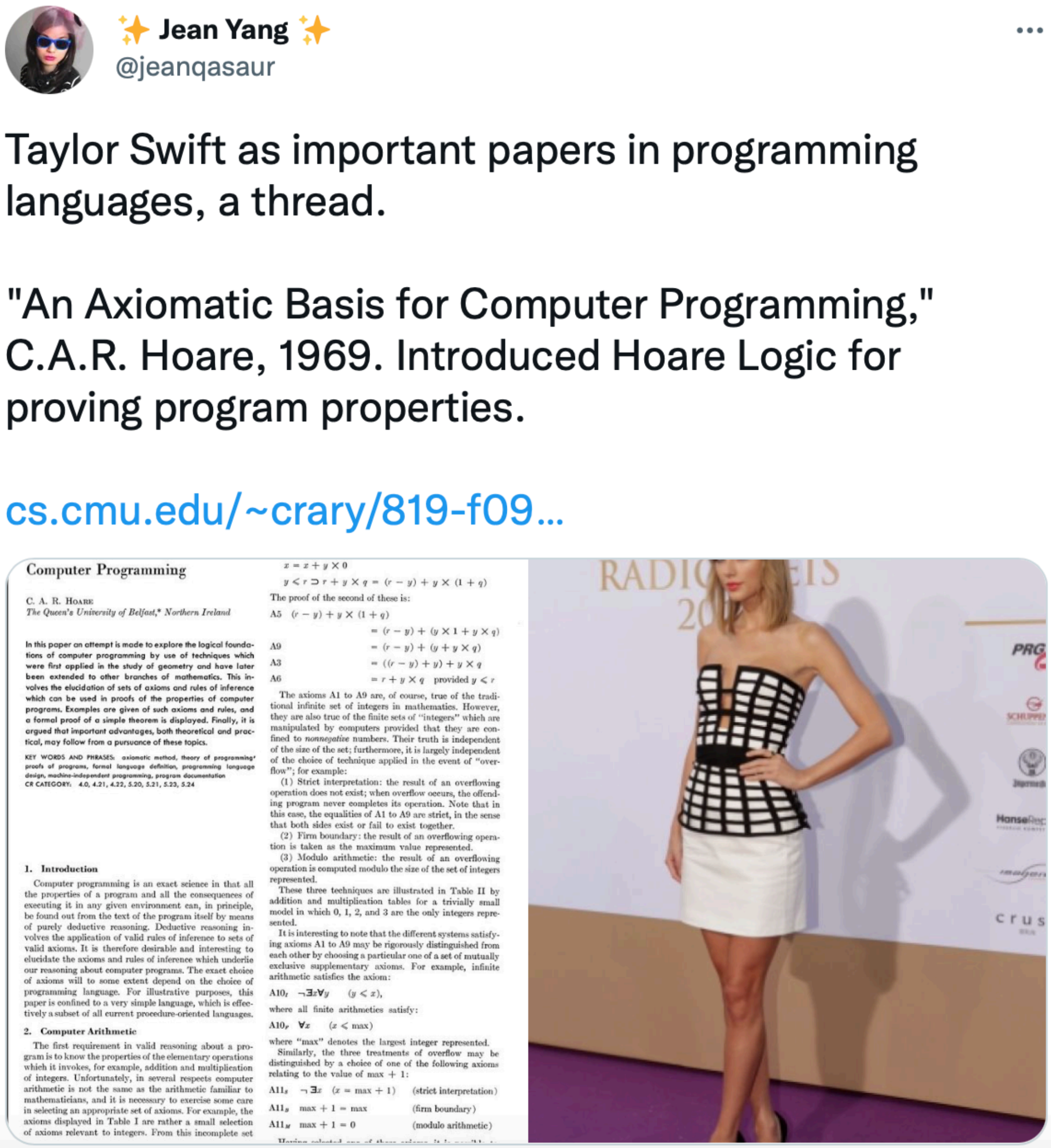
# … but …



Keep moving, even if slowly!

# Find your style

# Find your style



✨ Jean Yang ✨
@jeanqasaur

Taylor Swift as important papers in programming languages, a thread.

"An Axiomatic Basis for Computer Programming," C.A.R. Hoare, 1969. Introduced Hoare Logic for proving program properties.

cs.cmu.edu/~crary/819-f09…

Computer Programming

C. A. R. Hoare
*The Queen's University of Belfast,* Northern Ireland*

In this paper an attempt is made to explore the logical foundations of computer programming by use of techniques which were first applied in the study of geometry and have later been extended to other branches of mathematics. This involves the elucidation of sets of axioms and rules of inference which can be used in proofs of the properties of computer programs. Examples are given of such axioms and rules, and a formal proof of a simple theorem is displayed. Finally, it is argued that important advantages, both theoretical and practical, may follow from a pursuance of these topics.

KEY WORDS AND PHRASES: axiomatic method, theory of programming' proofs of programs, formal language definition, programming language design, machine-independent programming, program documentation
CR CATEGORY: 4.0, 4.21, 4.22, 5.20, 5.21, 5.23, 5.24

1. **Introduction**

Computer programming is an exact science in that all the properties of a program and all the consequences of executing it in any given environment can, in principle, be found out from the text of the program itself by means of purely deductive reasoning. Deductive reasoning involves the application of valid rules of inference to sets of valid axioms. It is therefore desirable and interesting to elucidate the axioms and rules of inference which underlie our reasoning about computer programs. The exact choice of axioms will to some extent depend on the choice of programming language. For illustrative purposes, this paper is confined to a very simple language, which is effectively a subset of all current procedure-oriented languages.

2. **Computer Arithmetic**

The first requirement in valid reasoning about a program is to know the properties of the elementary operations which it invokes, for example, addition and multiplication of integers. Unfortunately, in several respects computer arithmetic is not the same as the arithmetic familiar to mathematicians, and it is necessary to exercise some care in selecting an appropriate set of axioms. For example, the axioms displayed in Table I are rather a small selection of axioms relevant to integers. From this incomplete set

$x = x + y \times 0$

$y < r \supset r + y \times q = (r - y) + y \times (1 + q)$

The proof of the second of these is:

$$\begin{aligned}
&\text{A5} \quad (r - y) + y \times (1 + q) \\
&\qquad = (r - y) + (y \times 1 + y \times q) \\
&\text{A9} \qquad = (r - y) + (y + y \times q) \\
&\text{A3} \qquad = ((r - y) + y) + y \times q \\
&\text{A6} \qquad = r + y \times q \quad \text{provided } y < r
\end{aligned}$$

The axioms A1 to A9 are, of course, true of the traditional infinite set of integers in mathematics. However, they are also true of the finite sets of "integers" which are manipulated by computers provided that they are confined to *nonnegative* numbers. Their truth is independent of the size of the set; furthermore, it is largely independent of the choice of technique applied in the event of "overflow"; for example:

(1) Strict interpretation: the result of an overflowing operation does not exist; when overflow occurs, the offending program never completes its operation. Note that in this case, the equalities of A1 to A9 are strict, in the sense that both sides exist or fail to exist together.

(2) Firm boundary: the result of an overflowing operation is taken as the maximum value represented.

(3) Modulo arithmetic: the result of an overflowing operation is computed modulo the size of the set of integers represented.

These three techniques are illustrated in Table II by addition and multiplication tables for a trivially small model in which 0, 1, 2, and 3 are the only integers represented.

It is interesting to note that the different systems satisfying axioms A1 to A9 may be rigorously distinguished from each other by choosing a particular one of a set of mutually exclusive supplementary axioms. For example, infinite arithmetic satisfies the axiom:

A10$_I$ $\neg\exists x \forall y \quad (y \leqslant x)$,

where all finite arithmetics satisfy:

A10$_F$ $\forall x \quad (x \leqslant \max)$

where "max" denotes the largest integer represented.

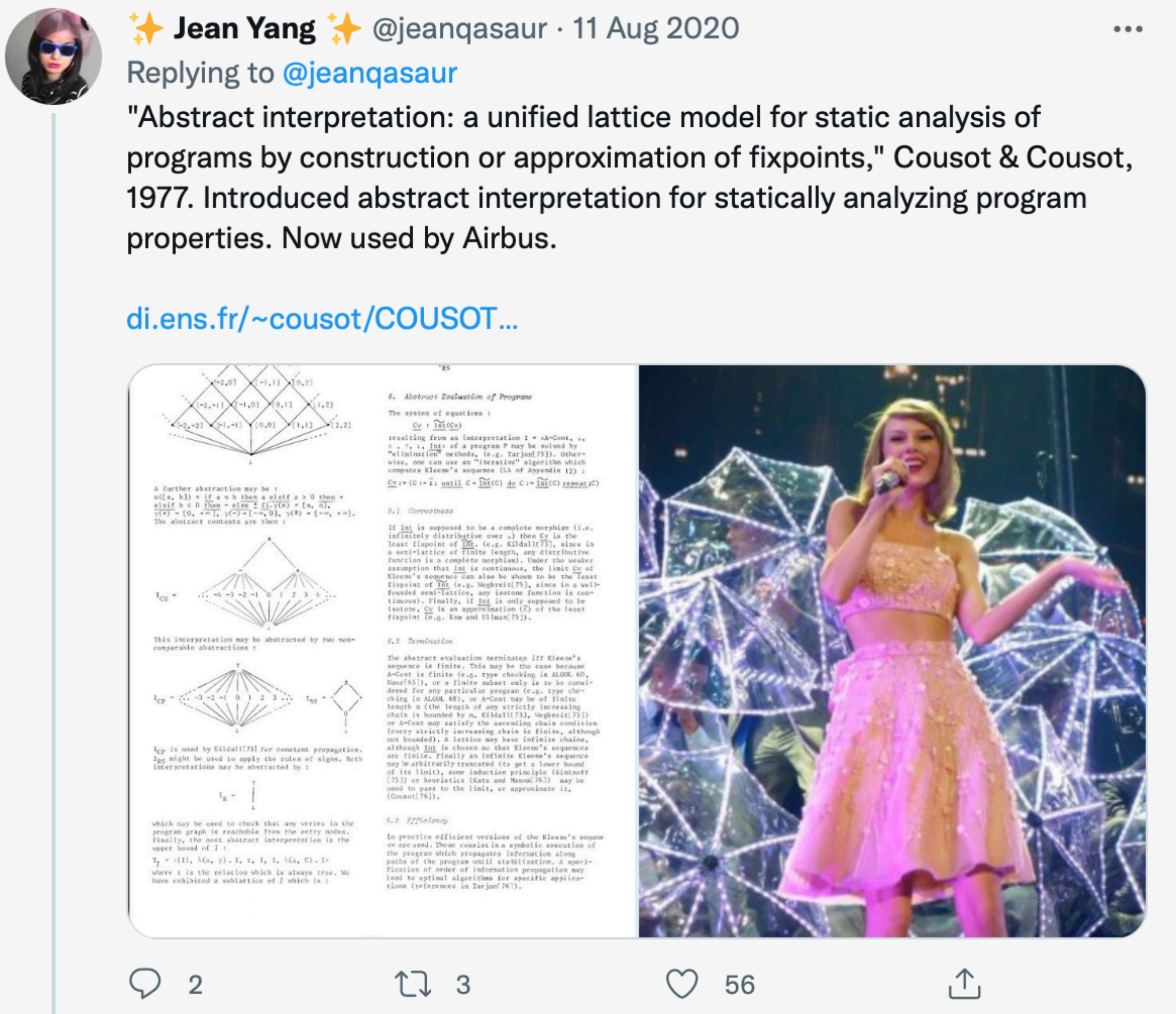
Similarly, the three treatments of overflow may be distinguished by a choice of one of the following axioms relating to the value of max + 1:

A11$_S$ $\neg\exists x \quad (x = \max + 1)$ (strict interpretation)

A11$_B$ $\max + 1 = \max$ (firm boundary)

A11$_M$ $\max + 1 = 0$ (modulo arithmetic)

# Find your style

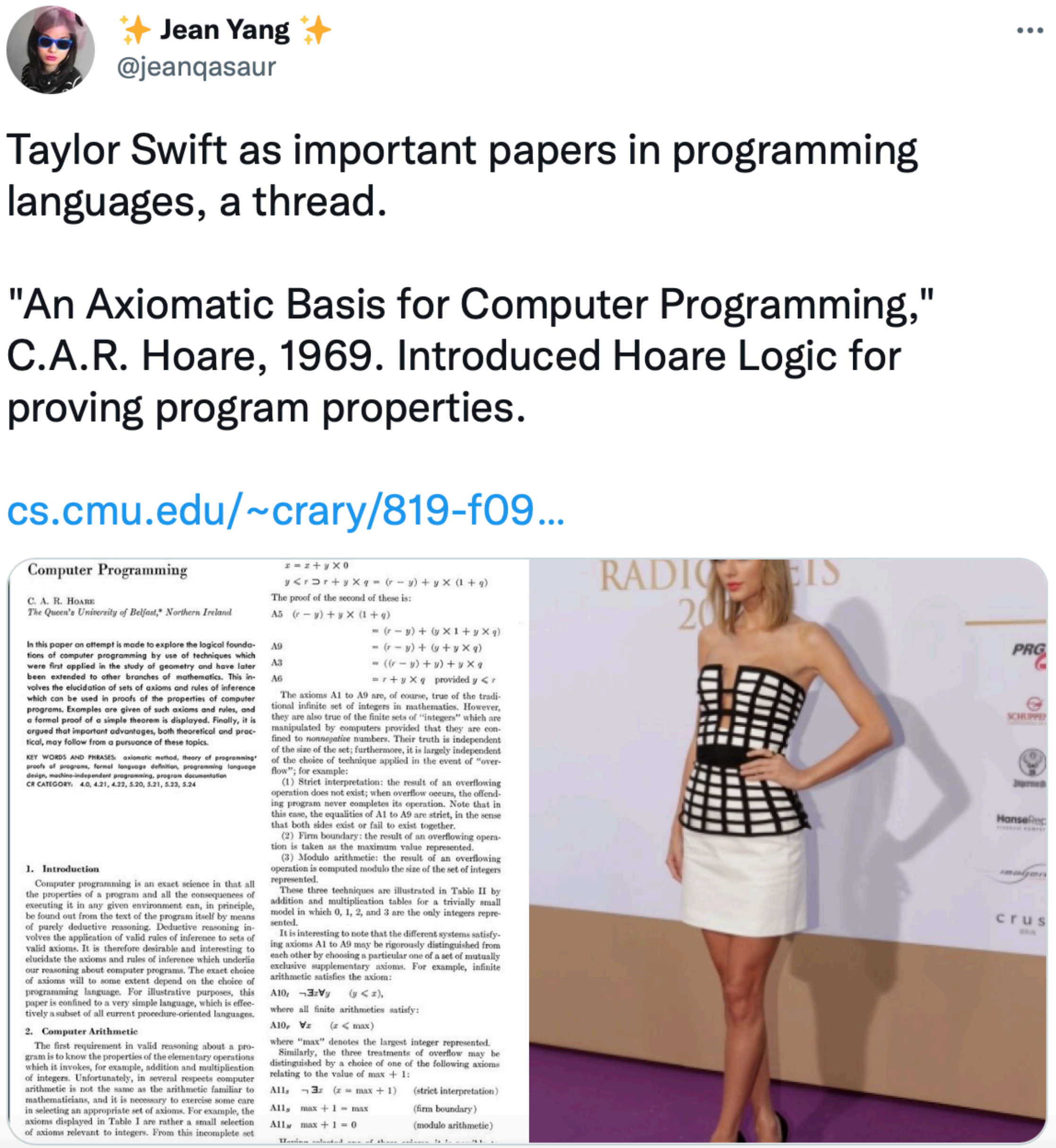


✨ Jean Yang ✨ @jeanqasaur · 11 Aug 2020

Replying to @jeanqasaur

"Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints," Cousot & Cousot, 1977. Introduced abstract interpretation for statically analyzing program properties. Now used by Airbus.

di.ens.fr/~cousot/COUSOT...



✨ Jean Yang ✨ @jeanqasaur

Taylor Swift as important papers in programming languages, a thread.

"An Axiomatic Basis for Computer Programming," C.A.R. Hoare, 1969. Introduced Hoare Logic for proving program properties.

cs.cmu.edu/~crary/819-f09...

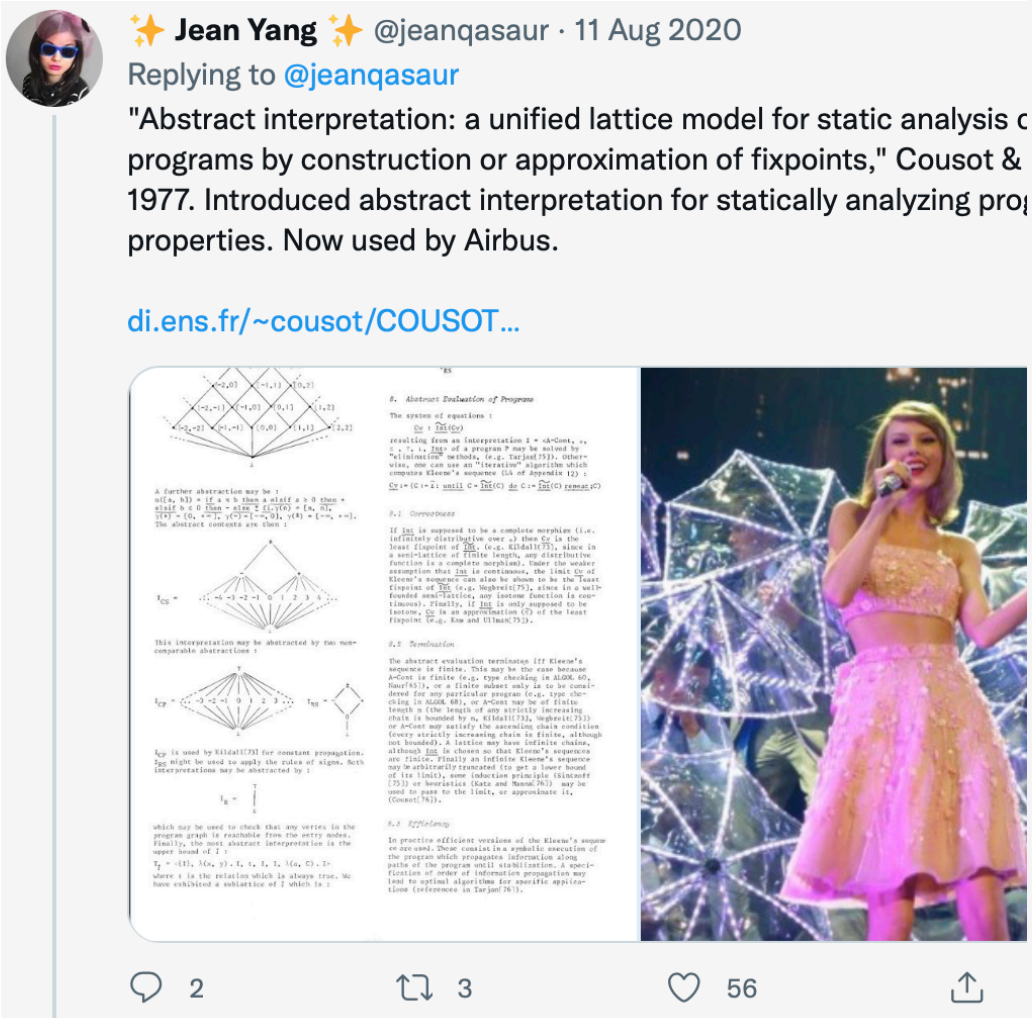
Computer Programming

C. A. R. Hoare
*The Queen's University of Belfast,* Northern Ireland*

In this paper an attempt is made to explore the logical foundations of computer programming by use of techniques which were first applied in the study of geometry and have later been extended to other branches of mathematics. This involves the elucidation of sets of axioms and rules of inference which can be used in proofs of the properties of computer programs. Examples are given of such axioms and rules, and a formal proof of a simple theorem is displayed. Finally, it is argued that important advantages, both theoretical and practical, may follow from a pursuance of these topics.

KEY WORDS AND PHRASES: axiomatic method, theory of programming' proofs of programs, formal language definition, programming language design, machine-independent programming, program documentation
CR CATEGORY: 4.0, 4.21, 4.22, 5.20, 5.21, 5.23, 5.24

# Find your style



✨ Jean Yang ✨ @jeanqasaur · 11 Aug 2020

Replying to @jeanqasaur

"Abstract interpretation: a unified lattice model for static analysis c
programs by construction or approximation of fixpoints," Cousot &
1977. Introduced abstract interpretation for statically analyzing pro
properties. Now used by Airbus.

di.ens.fr/~cousot/COUSOT...

8. Abstract Evaluation of Programs

The system of equations :

Cv : $\widetilde{\text{Int}}$(Cv)

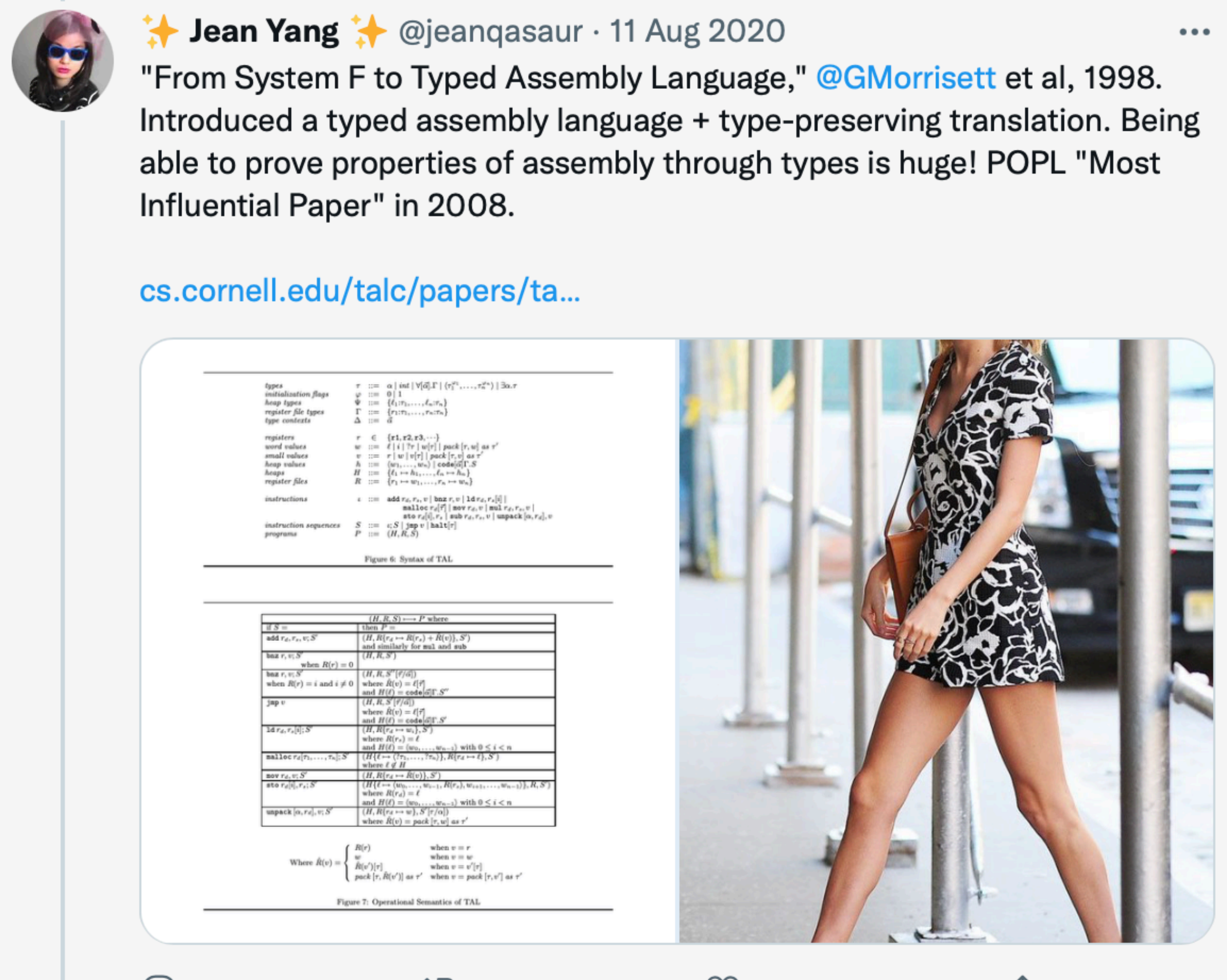
resulting from an interpretation I = <A-Cont, ∘, ≤ , ⊤, ⊥, Int> of a program P may be solved by "elimination" methods, (e.g. Tarjan[75]). Otherwise, one can use an "iterative" algorithm which computes Kleene's sequence (l.4 of Appendix 12) :

Cv := (C := ⊥̃; until C = $\widetilde{\text{Int}}$(C) do C := $\widetilde{\text{Int}}$(C) repeat;C)

8.1 Correctness

8.2 Termination

8.3 Efficiency



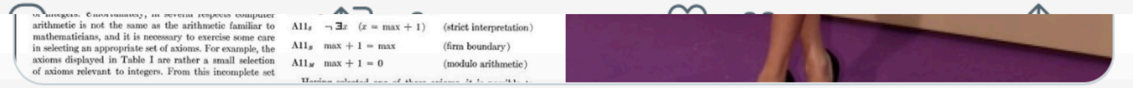
✨ Jean Yang ✨ @jeanqasaur · 11 Aug 2020

"From System F to Typed Assembly Language," @GMorrisett et al, 1998. Introduced a typed assembly language + type-preserving translation. Being able to prove properties of assembly through types is huge! POPL "Most Influential Paper" in 2008.

cs.cornell.edu/talc/papers/ta...

Figure 6: Syntax of TAL

Figure 7: Operational Semantics of TAL

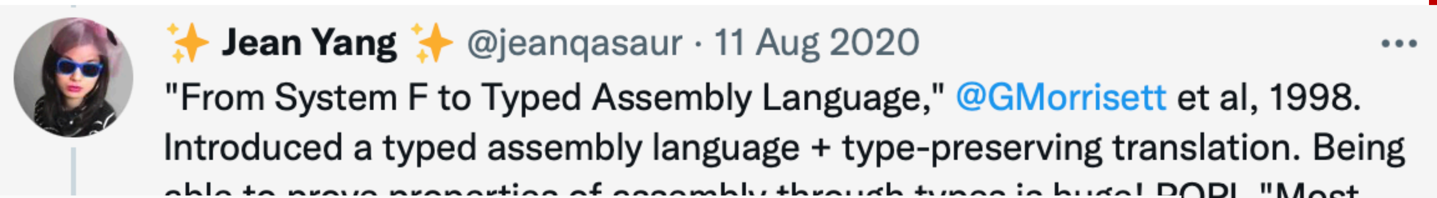


arithmetic is not the same as the arithmetic familiar to mathematicians, and it is necessary to exercise some care in selecting an appropriate set of axioms. For example, the axioms displayed in Table I are rather a small selection of axioms relevant to integers. From this incomplete set

| | | |
|---|---|---|
| $\mathbf{A11}_S$ | $\neg\exists x\ (x = \max + 1)$ | (strict interpretation) |
| $\mathbf{A11}_B$ | $\max + 1 = \max$ | (firm boundary) |
| $\mathbf{A11}_M$ | $\max + 1 = 0$ | (modulo arithmetic) |

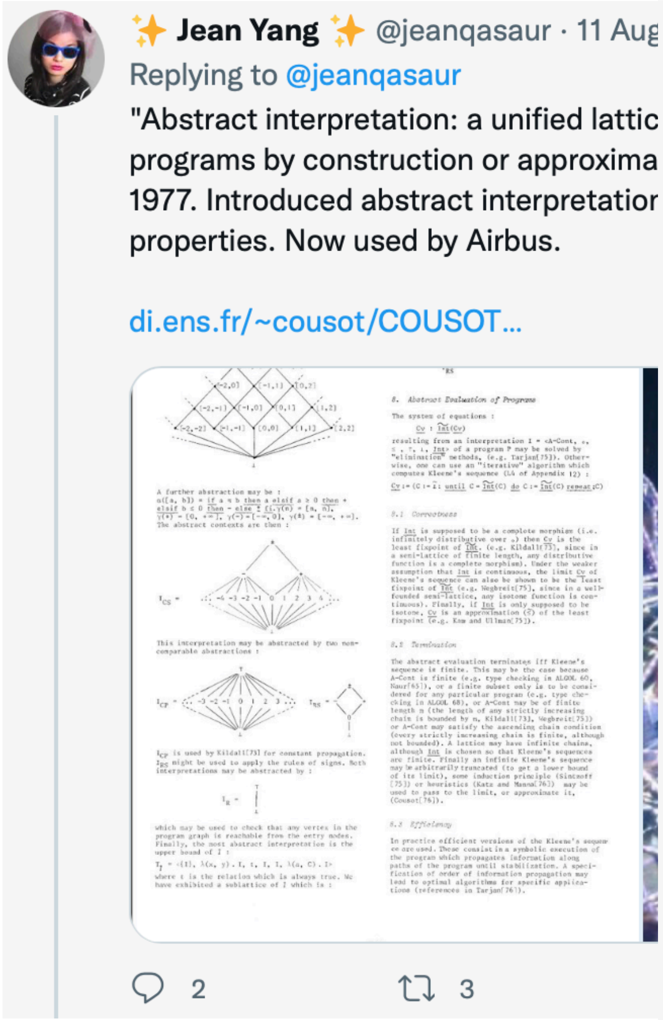
Cornell Bowers C·IS
Computer Science

# Find your style



✨ Jean Yang ✨ @jeanqasaur · 11 Aug 2020

"From System F to Typed Assembly Language," @GMorrisett et al, 1998. Introduced a typed assembly language + type-preserving translation. Being able to prove properties of assembly through types is huge! POPL "Most



✨ Jean Yang ✨ @jeanqasaur · 11 Aug

Replying to @jeanqasaur

"Abstract interpretation: a unified lattic programs by construction or approxima 1977. Introduced abstract interpretatior properties. Now used by Airbus.

di.ens.fr/~cousot/COUSOT...

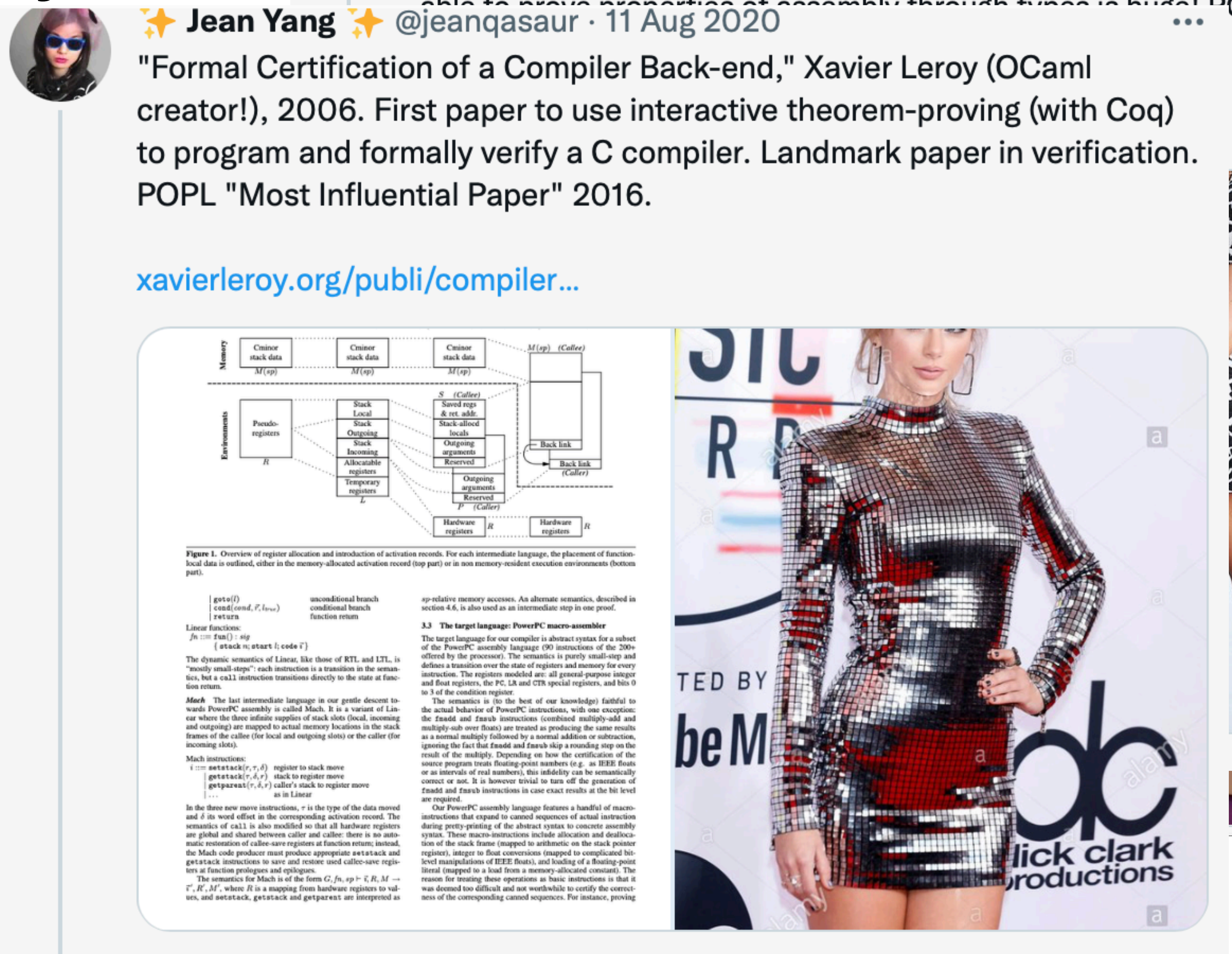
8. Abstract Evaluation of Programs

8.1 Correctness

8.2 Termination

8.3 Efficiency

2 3



✨ Jean Yang ✨ @jeanqasaur · 11 Aug 2020

"Formal Certification of a Compiler Back-end," Xavier Leroy (OCaml creator!), 2006. First paper to use interactive theorem-proving (with Coq) to program and formally verify a C compiler. Landmark paper in verification. POPL "Most Influential Paper" 2016.

xavierleroy.org/publi/compiler...

**Figure 1.** Overview of register allocation and introduction of activation records. For each intermediate language, the placement of function-local data is outlined, either in the memory-allocated activation record (top part) or in non memory-resident execution environments (bottom part).

3.3 The target language: PowerPC macro-assembler

# Find your style

# Find your style

Why is style important?

# More practically…

# More practically…

Read a lot (of introductions!)

# More practically…

Read a lot (of introductions!)

YouTube

# More practically…

Read a lot (of introductions!)

YouTube

Talk to people whose work you like

# More practically…

Read a lot (of introductions!)

Talk to peers

YouTube

Talk to people whose work you like

# More practically…

Read a lot (of introductions!)

Talk to peers

YouTube

Twitter

Talk to people whose work you like

# And when things go wrong?

# And when things go wrong?

You do not like the topic

# And when things go wrong?

You do not like the topic

Collaboration breakdown

# And when things go wrong?

You do not like the topic

Collaboration breakdown

Your work keeps getting rejected

# And when things go wrong?

You do not like the topic

Supervisor conflict

Collaboration breakdown

Your work keeps getting rejected

# And when things go wrong?

You do not like the topic

Supervisor conflict

Collaboration breakdown

Mental health

Your work keeps getting rejected

# Some reflections…

Finding a matching topic is important — for your daily happiness and for your career

# Some reflections…

Finding a matching topic is important — for your daily happiness and for your career

# Some reflections…

Finding a matching topic is important — for your daily happiness and for your career

Multidimensional decision
External Factors

# Some reflections…

Finding a matching topic is important — for your daily happiness and for your career

Multidimensional decision
External Factors

A topic is not forever

# Some reflections…

Finding a matching topic is important — for your daily happiness and for your career

Multidimensional decision
External Factors

Your research evolves with the years — topic-wise and style-wise

A topic is not forever

# Some reflections…

Finding a matching topic is important — for your daily happiness and for your career

Multidimensional decision
External Factors

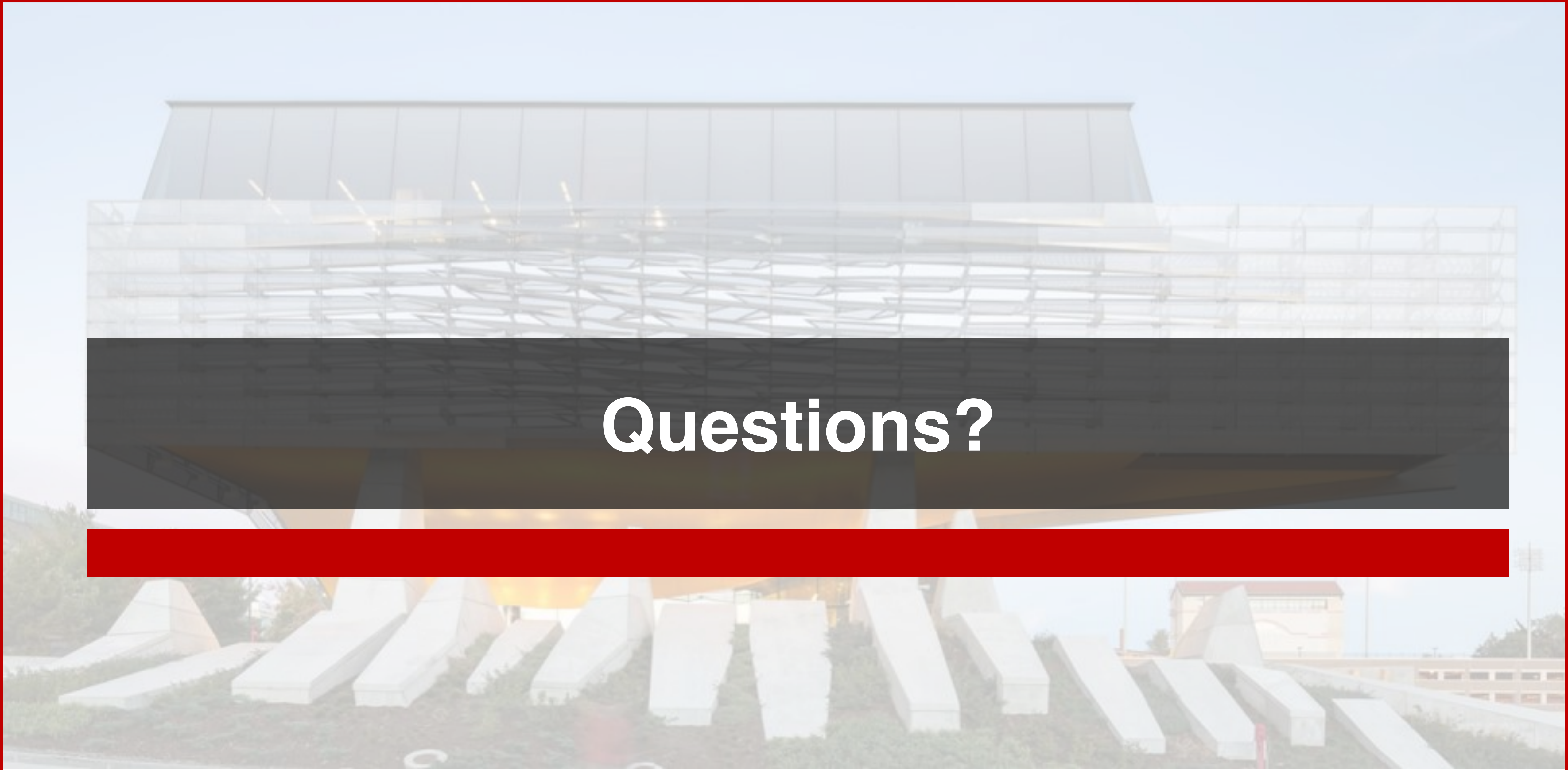
Your research evolves with the years — topic-wise and style-wise

A topic is not forever

Sometimes the topic finds you

# Research is a conversation

*- Stephanie Weirich, PLMW’22*



# Questions?