Nominal Representation of Syntax in Proof Assistants

Yee-Jian Tan
DistriNet, KU Leuven

Proof Assistant & Formal Verification

Mathematics formalisation is scaling quickly:

Fermat's Last Theorem, Navier-Stokes* | Feit-Thompson, Four colour, ...

Proof Assistant & Formal Verification

Mathematics formalisation is scaling quickly:

Fermat's Last Theorem, Navier-Stokes* | Feit-Thompson, Four colour, ...

E. W. Dijkstra: *"Program testing can be used to show the presence of bugs, but never to show their absence!"*

Compiler (CompCert), OS (seL4), protocol (TLS), ...

Life-critical software and protocols

Proof Assistant & Formal Verification

Mathematics formalisation is scaling quickly:

Fermat's Last Theorem, Navier-Stokes* | Feit-Thompson, Four colour, ...

E. W. Dijkstra: *"Program testing can be used to show the presence of bugs, but never to show their absence!"*

Compiler (CompCert), OS (seL4), protocol (TLS), ...

Life-critical software and protocols

Type Theory: type-checking *is* proof-checking

Proof Assistant & Formal Verification

Mathematics formalisation is scaling quickly:

Fermat's Last Theorem, Navier-Stokes* | Feit-Thompson, Four colour, ...

E. W. Dijkstra: *"Program testing can be used to show the presence of bugs, but never to show their absence!"*

Compiler (CompCert), OS (seL4), protocol (TLS), ...

Life-critical software and protocols

Type Theory: type-checking *is* proof-checking

What makes formal verification of *programs* **challenging**?

Programming Language (PL) Meta-theory

What do we “mean” with a program?

"Semantics" meaning

⇳ meta-theory

"Syntax" program

Programming Language (PL) Meta-theory

What do we “mean” with a program?

"Semantics" meaning

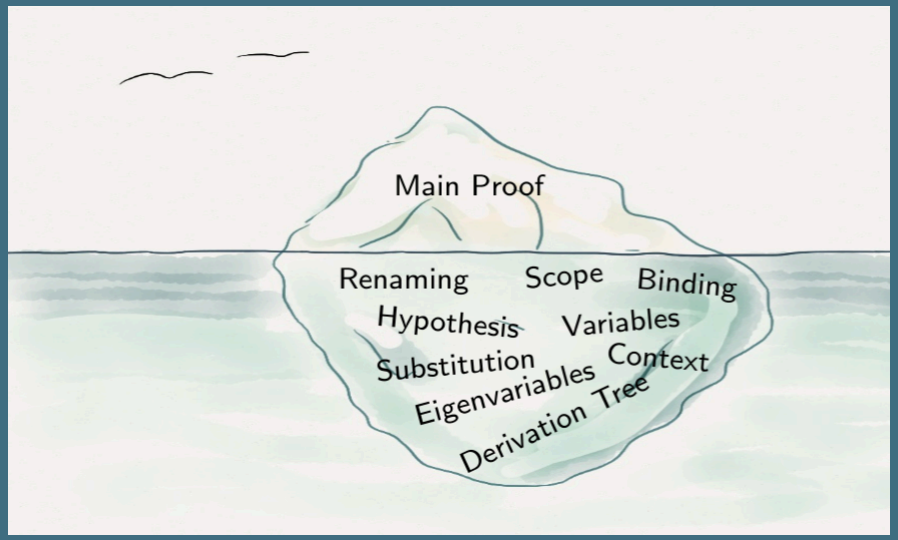
⇳ meta-theory

"Syntax" program

Meta-theory is **difficult**.

2005: PoplMark: formalising PL meta-theory [Ayd+05]

2019: PoplMark Reloaded: new benchmark problems [Abe+19]



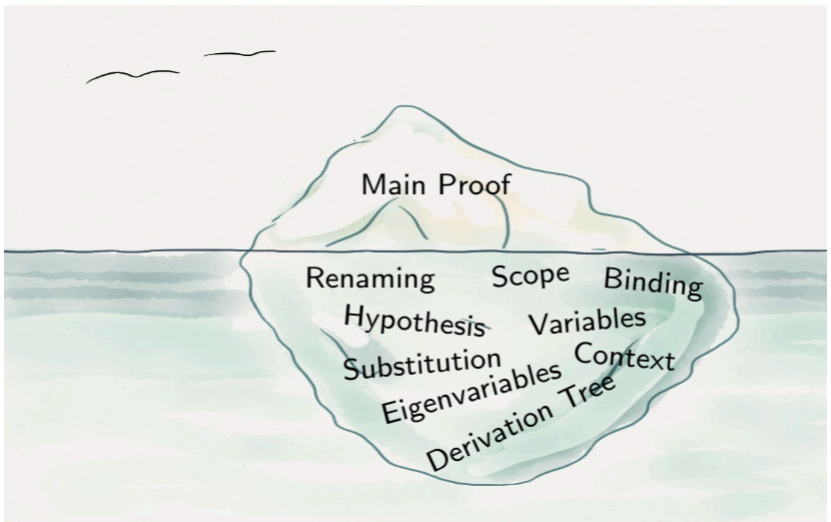
We may think of [the] proof as an iceberg. In the top of it, we find what we usually consider the real proof; underwater, the most of the matter, consisting of all mathematical preliminaries a reader must know in order to understand what is going on.

— S. Berardi [1990]

Syntax with Binders is Hard

2005: PoplMark challenge [Ayd+05]

2019: PoplMark Reloaded[Abe+19]



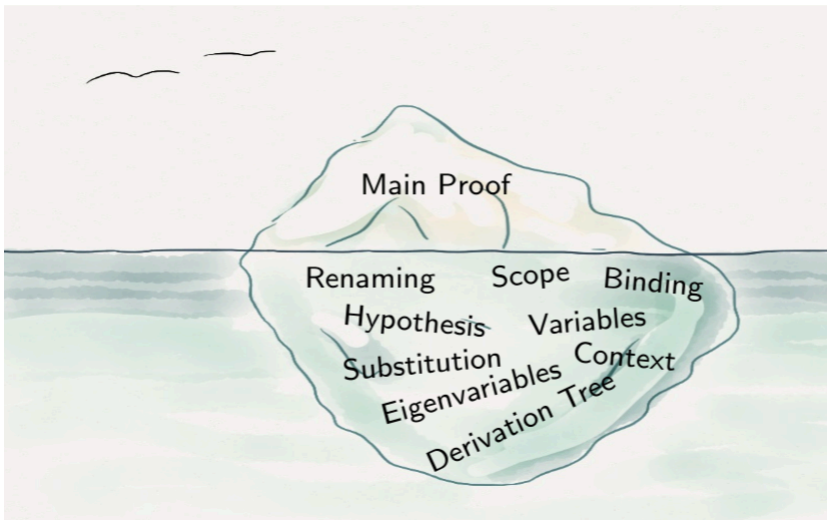
Source: “Mechanising Meta-Theory in Beluga” by Brigitte Pientka

Syntax with Binders is Hard

2005: PoplMark challenge [Ayd+05]:

one of the **"most critical issues"** in formalizing PL meta-theory

2019: PoplMark Reloaded[Abe+19]



Source: "Mechanising Meta-Theory in Beluga" by Brigitte Pientka

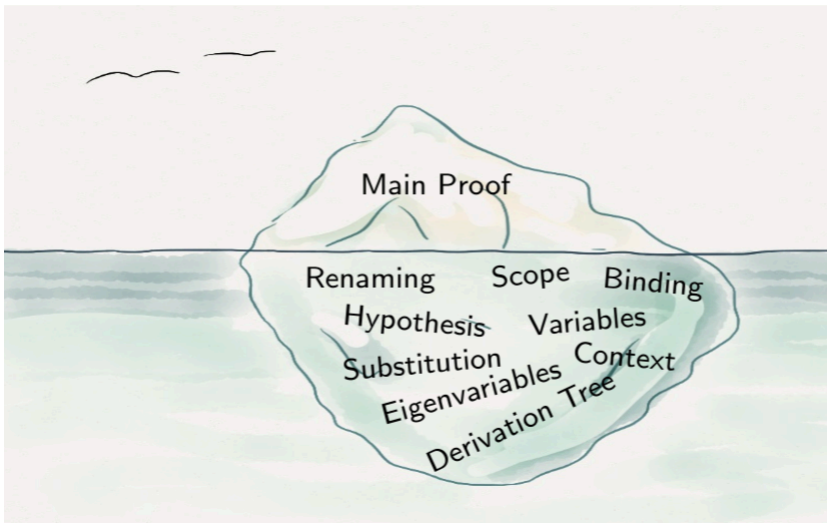
Syntax with Binders is Hard

2005: PoplMark challenge [Ayd+05]:

one of the **"most critical issues"** in formalizing PL meta-theory

2019: PoplMark Reloaded[Abe+19]:

"*costs/benefits* of various **variable binding representation**"



Source: "Mechanising Meta-Theory in Beluga" by Brigitte Pientka

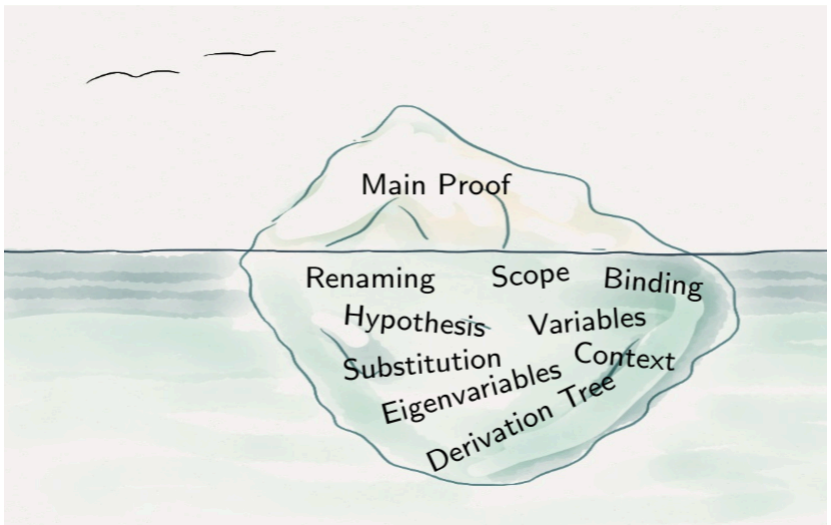
Syntax with Binders is Hard

2005: PoplMark challenge [Ayd+05]:

one of the **"most critical issues"** in formalizing PL meta-theory

2019: PoplMark Reloaded[Abe+19]:

"*costs/benefits* of various **variable binding representation**"



Source: "Mechanising Meta-Theory in Beluga" by Brigitte Pientka

What is a **good** variable binding representation?

Syntax with Binders: Criteria

```C
int a;
int f (int x) {
  return x + a;
}
```

Syntax with Binders: Criteria

```C
1  int a;
2  int f (int x) {
3    return x + a;
4  }
```

```C
1  int a;
2  int f (int y) {
3    return y + a;
4  }
```

- α-equivalence

Syntax with Binders: Criteria

```C
1  int a;
2  int f (int x) {
3    return x + a;
4  }
```

```C
1  int a;
2  int f (int a) {
3    return a + a;
4  }
```

- α-equivalence
- capture-avoidance

Syntax with Binders: Criteria

```c
int a;
int f (int x) {
  return x + a;
}
```

$$\lambda.v_0 + v_1$$

- α-equivalence
- capture-avoidance
- readability

Syntax with Binders: Criteria

```C
int a;
int f (int x) {
  return x + a;
}
```

Induction:

Assume the binding variable is “z” ...

- α-equivalence
- capture-avoidance
- readability
- binder-induction

Syntax with binders: Comparison

	α-equivalence	capture-avoidance	readability	binder-induction	TT Impl?
Named Vars	✗	✗	✓	✗	✓
de Bruijn	✓	✓	✗	✓	✓
HOAS	✓	✓	✓	✗	✓

Syntax with binders: Comparison

	α-equivalence	capture-avoidance	readability	binder-induction	TT Impl?
Named Vars	✗	✗	✓	✗	✓
de Bruijn	✓	✓	✗	✓	✓
HOAS	✓	✓	✓	✗	✓
Nominal	✓	✓	✓	✓	

Syntax with binders: Comparison

	α-equivalence	capture-avoidance	readability	binder-induction	TT Impl?
Named Vars	✗	✗	✓	✗	✓
de Bruijn	✓	✓	✗	✓	✓
HOAS	✓	✓	✓	✗	✓
Nominal	✓	✓	✓	✓	✗

A **Nominal TT** is needed!

Nominal TT

Breakthrough: *Nullary Parametric* TT realises **Nominal** TT [VND25].

Parametricity: "Representation Independence" [Wad89]

WP1. Implement and Evaluate **Nominal Agda**
WP2. Develop **Nominal Observational TT** (NOTT)
WP3. Implement NOTT, **unifying** with other TT variants for maximal dissemination

Yee-Jian Tan

Education

🇸🇬 NUS (QS 🔟): Math + CS, Hons

🇫🇷 MPRI: theoretical CS research masters

🇧🇪 KU Leuven: DistriNet (TT subgroup)

Awards

🏆 *Mention de Félicitations* for L3/M1 Theses

🪙 Full scholarship (bachelors, masters)

Research

🤝 1 invited workshop talk

🌏 7 int'l conference/symposium (3 talks 🔈)

📋 1 peer-reviewed publication
1 in review, 1 planned

🔧 4 workshop talks

Research Group

🖥️ DistriNet @ KU Leuven:
Hardware ↔ Software ↔ Logic

👥 D. Devriese, A. Nuyts:

- only **Prf. Assistant and (Dep)TT** subgroup in 🇧🇪
- organised Agda Implementor's Meeting (Apr 2026)

People

Agda Core Dev Team (A. Abel, J. Cockx)
Rocq Core Dev Team (INRIA Gallinette)
INRIA Cambium, Budapest TT Group, ...

Teaching

👤 Intern (Miguel Calçado, UMinho)

👤 Master's Thesis (Ridan Vandenbergh, KU Leuven)

🏆 Teaching Assistant (Honour List of Student Tutors)

References

[Ayd+05] B. E. Aydemir *et al.*, “Mechanized Metatheory for the Masses: The PoplMark Challenge,” in *Theorem Proving in Higher Order Logics, 18th International Conference, TPHOLs 2005, Oxford, UK, August 22-25, 2005, Proceedings*, J. Hurd and T. F. Melham, Eds., in Lecture Notes in Computer Science, vol. 3603. Springer, 2005, pp. 50–65. doi: 10.1007/11541868_4.

[Abe+19] A. Abel *et al.*, “POPLMark reloaded: Mechanizing proofs by logical relations,” *Journal of Functional Programming*, vol. 29, p. e19, 2019, doi: 10.1017/S0956796819000170.

[VND25] A. Van Muylder, A. Nuyts, and D. Devriese, “Nominal Type Theory by Nullary Internal Parametricity.” [Online]. Available: https://arxiv.org/abs/2512.09464

[Wad89] P. Wadler, “Theorems for Free!,” in *FPCA '89*, Imperial College, London, United Kingdom: ACM, 1989, pp. 347–359. doi: 10.1145/99370.99404.