What is Dependent Type Theory and why does Lean use it?

Lino Haupt

Outline

Introduction

Introduction

Motivation: Why Type Theory?

- Lean is a *proof verifier* **and** a *proof assistant*.
- Its kernel is designed to be:
  - **Technically simple** (avoid bugs in the implementation).
  - **Theoretically understandable** (metatheory is well-understood).
  - **Mathematically sound** (no paradoxes).
- **Goal of this talk**: Explain why Lean uses **dependent type theory (DTT)** instead of first-order logic (FOL).

Introduction

Motivation: Why Type Theory?

- Lean is a *proof verifier* **and** a *proof assistant*.
- Its kernel is designed to be:
  - **Technically simple** (avoid bugs in the implementation).
  - **Theoretically understandable** (metatheory is well-understood).
  - **Mathematically sound** (no paradoxes).
- **Goal of this talk**: Explain why Lean uses **dependent type theory (DTT)** instead of first-order logic (FOL).

Introduction

Motivation: Why Type Theory?

- Lean is a *proof verifier* **and** a *proof assistant*.
- Its kernel is designed to be:
  - **Technically simple** (avoid bugs in the implementation).
  - **Theoretically understandable** (metatheory is well-understood).
  - **Mathematically sound** (no paradoxes).
- **Goal of this talk**: Explain why Lean uses **dependent type theory (DTT)** instead of first-order logic (FOL).

Introduction

Motivation: Why Type Theory?

- Lean is a *proof verifier* **and** a *proof assistant*.
- Its kernel is designed to be:
  - **Technically simple** (avoid bugs in the implementation).
  - **Theoretically understandable** (metatheory is well-understood).
  - **Mathematically sound** (no paradoxes).
- **Goal of this talk**: Explain why Lean uses **dependent type theory (DTT)** instead of first-order logic (FOL).

Introduction

Motivation: Why Type Theory?

- Lean is a *proof verifier* **and** a *proof assistant*.
- Its kernel is designed to be:
  - **Technically simple** (avoid bugs in the implementation).
  - **Theoretically understandable** (metatheory is well-understood).
  - **Mathematically sound** (no paradoxes).
- **Goal of this talk:** Explain why Lean uses **dependent type theory (DTT)** instead of first-order logic (FOL).

Introduction

Motivation: Why Type Theory?

- Lean is a *proof verifier* **and** a *proof assistant*.
- Its kernel is designed to be:
  - **Technically simple** (avoid bugs in the implementation).
  - **Theoretically understandable** (metatheory is well-understood).
  - **Mathematically sound** (no paradoxes).
- **Goal of this talk**: Explain why Lean uses **dependent type theory (DTT)** instead of first-order logic (FOL).

Introduction

In what ways does Lean try to help?

- Being able to check equality by reducing to normal forms.
- Being able to check if certain theorems fit in a gap in an attempted proof.
- Tell you about the remaining goals in the proof.
- Give you some information about the objects you reference.
- Give you some information about the hypotheses you use.
- many more...

Introduction

In what ways does Lean try to help?

- Being able to check equality by reducing to normal forms.
- Being able to check if certain theorems fit in a gap in an attempted proof.
- Tell you about the remaining goals in the proof.
- Give you some information about the objects you reference.
- Give you some information about the hypotheses you use.
- many more...

Introduction

In what ways does Lean try to help?

- Being able to check equality by reducing to normal forms.
- Being able to check if certain theorems fit in a gap in an attempted proof.
- Tell you about the remaining goals in the proof.
- Give you some information about the objects you reference.
- Give you some information about the hypotheses you use.
- many more...

Introduction

In what ways does Lean try to help?

- Being able to check equality by reducing to normal forms.
- Being able to check if certain theorems fit in a gap in an attempted proof.
- Tell you about the remaining goals in the proof.
- Give you some information about the objects you reference.
- Give you some information about the hypotheses you use.
- many more...

Introduction

In what ways does Lean try to help?

- Being able to check equality by reducing to normal forms.
- Being able to check if certain theorems fit in a gap in an attempted proof.
- Tell you about the remaining goals in the proof.
- Give you some information about the objects you reference.
- Give you some information about the hypotheses you use.
- many more...

Introduction

In what ways does Lean try to help?

- Being able to check equality by reducing to normal forms.
- Being able to check if certain theorems fit in a gap in an attempted proof.
- Tell you about the remaining goals in the proof.
- Give you some information about the objects you reference.
- Give you some information about the hypotheses you use.
- many more...

Why not First Order Logic and Set Theory?

Mathematics and Set Theory

"Mathematics is based on set theory as a foundation" is a common claim. But in what sense is "foundation" and "based" to be understood?

Mathematics and Set Theory

Under some meanings of “foundation” the claim seems **right**:

- Encode all[1] mathematical objects as sets.
- Allows for drastic reduction of complexity in foundations: Consistency, completeness, existence of models etc. can be studied abstractly. For example: Set theory allows us to see that the theories of Peano-Arithmetic and even of real and complex numbers are consistent (relative to ZFC).
- Provides a common language to show existence: You do not have to argue philosophically, just construct it.

[1] except for issues with collections too large, e. g. some categories

Mathematics and Set Theory

Under some meanings of “foundation” the claim seems **right**:

- Encode all[1] mathematical objects as sets.
- Allows for drastic reduction of complexity in foundations:
  Consistency, completeness, existence of models etc. can be studied abstractly. For example: Set theory allows us to see that the theories of Peano-Arithmetic and even of real and complex numbers are consistent (relative to ZFC).
- Provides a common language to show existence:
  You do not have to argue philosophically, just construct it.

[1] except for issues with collections too large, e. g. some categories

Mathematics and Set Theory

Under some meanings of “foundation” the claim seems **right**:

- Encode all[1] mathematical objects as sets.
- Allows for drastic reduction of complexity in foundations:
  Consistency, completeness, existence of models etc. can be studied abstractly. For example: Set theory allows us to see that the theories of Peano-Arithmetic and even of real and complex numbers are consistent (relative to ZFC).
- Provides a common language to show existence:
  You do not have to argue philosophically, just construct it.

[1] except for issues with collections too large, e. g. some categories

Mathematics and Set Theory

Under some meanings of “foundation” the claim seems **right**:

- Encode all[1] mathematical objects as sets.
- Allows for drastic reduction of complexity in foundations: Consistency, completeness, existence of models etc. can be studied abstractly. For example: Set theory allows us to see that the theories of Peano-Arithmetic and even of real and complex numbers are consistent (relative to ZFC).
- Provides a common language to show existence: You do not have to argue philosophically, just construct it.

[1] except for issues with collections too large, e. g. some categories

Mathematics and Set Theory

Under some meanings of “foundation” the claim seems **right**:

- Encode all[1] mathematical objects as sets.
- Allows for drastic reduction of complexity in foundations: Consistency, completeness, existence of models etc. can be studied abstractly. For example: Set theory allows us to see that the theories of Peano-Arithmetic and even of real and complex numbers are consistent (relative to ZFC).
- Provides a common language to show existence: You do not have to argue philosophically, just construct it.

[1] except for issues with collections too large, e. g. some categories

Mathematics and Set Theory

Under some meanings of “foundation” the claim seems **right**:

- Encode all[1] mathematical objects as sets.
- Allows for drastic reduction of complexity in foundations: Consistency, completeness, existence of models etc. can be studied abstractly. For example: Set theory allows us to see that the theories of Peano-Arithmetic and even of real and complex numbers are consistent (relative to ZFC).
- Provides a common language to show existence: You do not have to argue philosophically, just construct it.

[1] except for issues with collections too large, e. g. some categories

Mathematics and Set Theory

Under some meanings of "foundation" the claim seems **right**:

- Encode all[1] mathematical objects as sets.
- Allows for drastic reduction of complexity in foundations: Consistency, completeness, existence of models etc. can be studied abstractly. For example: Set theory allows us to see that the theories of Peano-Arithmetic and even of real and complex numbers are consistent (relative to ZFC).
- Provides a common language to show existence: You do not have to argue philosophically, just construct it.

[1] except for issues with collections too large, e. g. some categories

Mathematics and Set Theory

Under some meanings of "foundation" the claim seems **wrong**:

- "Junk-Theorems/Questions":[2]
  - Can a finite simple group be a zero of the Riemann Zeta Function?
  - Are the natural numbers a subset of the integers?
  - Is 1 an element of $(1, 3)$? Is 3 an element of $(1, 3)$?
- Mathematicians in practice use different sorts of objects:
  - Numbers
  - Sets
  - Functions
  - Vectorspaces
- We often do calculations without proving theorems about equality.

[2] See https://mathoverflow.net/questions/90820/set-theories-without-junk-theorems

Mathematics and Set Theory

Under some meanings of "foundation" the claim seems **wrong**:

- "Junk-Theorems/Questions":[2]
  - Can a finite simple group be a zero of the Riemann Zeta Function?
  - Are the natural numbers a subset of the integers?
  - Is 1 an element of $(1, 3)$? Is 3 an element of $(1, 3)$?
- Mathematicians in practice use different sorts of objects:
  - Numbers
  - Sets
  - Functions
  - Vectorspaces
- We often do calculations without proving theorems about equality.

[2] See https://mathoverflow.net/questions/90820/set-theories-without-junk-theorems

Mathematics and Set Theory

Under some meanings of "foundation" the claim seems **wrong**:

- "Junk-Theorems/Questions":[2]
  - Can a finite simple group be a zero of the Riemann Zeta Function?
  - Are the natural numbers a subset of the integers?
  - Is 1 an element of $(1, 3)$? Is 3 an element of $(1, 3)$?
- Mathematicians in practice use different sorts of objects:
  - Numbers
  - Sets
  - Functions
  - Vectorspaces
- We often do calculations without proving theorems about equality.

[2] See `https://mathoverflow.net/questions/90820/set-theories-without-junk-theorems`

Mathematics and Set Theory

Under some meanings of "foundation" the claim seems **wrong**:

- "Junk-Theorems/Questions":[2]
  - Can a finite simple group be a zero of the Riemann Zeta Function?
  - Are the natural numbers a subset of the integers?
  - Is 1 an element of $(1, 3)$? Is 3 an element of $(1, 3)$?
- Mathematicians in practice use different sorts of objects:
  - Numbers
  - Sets
  - Functions
  - Vectorspaces
- We often do calculations without proving theorems about equality.

[2] See `https://mathoverflow.net/questions/90820/set-theories-without-junk-theorems`

Mathematics and Set Theory

Conclusion

Set theory is a foundation of mathematics, as it allows universal encoding.

This universal encoding does not represent mathematical practice but is chosen for its technical merits in certain questions of meta-theory.

Mathematics and Set Theory

Conclusion

Set theory is a foundation of mathematics, as it allows universal encoding.
This universal encoding does not represent mathematical practice but is chosen for its technical merits in certain questions of meta-theory.

Example

Turing machines are also the foundation of the theory of computation:

- All programs can be encoded as a Turing machine.
- This allows to prove impossibility results etc.

But this foundation does neither reflect the practice of computation nor of programming:

- Computers do not function like Turing machines.
- We do not write programs in the language of Turing machines.

Example

Turing machines are also the foundation of the theory of computation:

- All programs can be encoded as a Turing machine.
- This allows to prove impossibility results etc.

But this foundation does neither reflect the practice of computation nor of programming:

- Computers do not function like Turing machines.
- We do not write programs in the language of Turing machines.

Example

Turing machines are also the foundation of the theory of computation:

- All programs can be encoded as a Turing machine.
- This allows to prove impossibility results etc.

But this foundation does neither reflect the practice of computation nor of programming:

- Computers do not function like Turing machines.
- We do not write programs in the language of Turing machines.

Example

Turing machines are also the foundation of the theory of computation:

- All programs can be encoded as a Turing machine.
- This allows to prove impossibility results etc.

But this foundation does neither reflect the practice of computation nor of programming:

- Computers do not function like Turing machines.
- We do not write programs in the language of Turing machines.

Conclusion:

We can sum this up in as:

Lean does not use First Order Logic for the same reason why we do not all write Code in `Assembly` and not even in C.

Or a bit more positive:

Lean's goal is to mirror mathematical practice, not to study set theory. Type theory is a better fit because it aligns with how we write proofs and structure ideas.

Now let us take a look at type theory to see if it indeed is better aligned.

Conclusion:

We can sum this up in as:

Lean does not use First Order Logic for the same reason why we do not all write Code in `Assembly` and not even in `C`.

Or a bit more positive:

Lean's goal is to mirror mathematical practice, not to study set theory. Type theory is a better fit because it aligns with how we write proofs and structure ideas.

Now let us take a look at type theory to see if it indeed is better aligned.

Conclusion:

We can sum this up in as:

Lean does not use First Order Logic for the same reason why we do not all write Code in `Assembly` and not even in `C`.

Or a bit more positive:

Lean's goal is to mirror mathematical practice, not to study set theory. Type theory is a better fit because it aligns with how we write proofs and structure ideas.

Now let us take a look at type theory to see if it indeed is better aligned.