A formal proof of the independence of the continuum hypothesis

Jesse Michael Han
Lean Together 2020
University of Pittsburgh
joint w/ Floris van Doorn

Outline

Introduction

Syntax

Forcing

Conclusions

Lean Together 2019

Towards a formal proof of the independence of the continuum hypothesis

Jesse Han

(joint with Floris van Doorn)

University of Pittsburgh

Lean Together 2019

Continuum hypothesis

- Posed by Cantor in 19th century: does there exist an infinite cardinality strictly larger than the countable natural numbers $\mathbb{N}$ but strictly smaller than the uncountable real numbers $\mathbb{R}$?

- was Hilbert's 1st question

- Proved independent (neither provable nor disprovable) from ZFC by Paul Cohen ('60s) and Kurt Godel ('30s). Cohen's invention of forcing earned him a Fields medal, the only one ever awarded for work in mathematical logic.

Continuum hypothesis

- Independence of CH had never been formalized

Formalizing 100 Theorems

There used to exist a "top 100" of mathematical theorems on the web, which is a rather arbitrary list (and most of the theorems seem rather elementary), but still is nice to look at. On the current page I will keep track of which theorems from this list have been formalized. Currently the fraction that already has been formalized seems to be

94%

24. *The Undecidability of the Continuum Hypothesis*

Continuum hypothesis

- Independence of CH had never been formalized... until now!

FLYPITCH

formally proving the independence of the continuum hypothesis

Website: flypitch.github.io

- Formalized the independence of CH
- Built reusable libraries for mathematical logic and set theory
- Written in Lean 3.

What is required for the formalization?

To formalize just the statement, "the continuum hypothesis is neither provable nor disprovable from ZFC", we need:

- Syntax: first-order logic (terms, formulas, quantifiers, sentences. . .)
- provability, i.e. a proof system
- the axioms of ZFC and also CH as first-order formulas

To formalize the proof, we need:

- Semantics (ordinary soundness theorem)
- Boolean-valued semantics and soundness for first-order logic
- Boolean-valued models of ZFC
- Forcing

First-order logic

```
structure Language : Type (u+1) :=
  (functions : ℕ → Type u)
  (relations : ℕ → Type u)

/- The language of abelian groups -/
inductive abel_functions : ℕ → Type
| zero : abel_functions 0
| plus : abel_functions 2

def L_abel : Language := ⟨abel_functions, (λ _, empty)⟩
```

First-order logic

```
inductive preterm : ℕ → Type u
| var : ∀ (k : ℕ), preterm 0 -- notation &
| func : ∀ {l : ℕ} (f : L.functions l), preterm l
| app : ∀ {l : ℕ} (t : preterm (l + 1)) (s : preterm 0),
    preterm l

def term := preterm L 0
```

- `preterm L n` is a partially applied term. If applied to `n` terms, it becomes a `term`.
- Every element of `preterm L 0` is a well-formed term.
- We use this encoding to avoid mutual or nested inductive types, since those are not too convenient to work with in Lean.

First-order logic

Similarly for formulas:

```
inductive preformula : ℕ → Type u
| falsum {} : preformula 0 -- notation ⊥
| equal (t₁ t₂ : term L) : preformula 0 -- notation ≃
| rel {l : ℕ} (R : L.relations l) : preformula l
| apprel {l : ℕ} (f : preformula (l + 1)) (t : term L) :
    preformula l
| imp (f₁ f₂ : preformula 0) : preformula 0 -- notation ⟹
| all (f : preformula 0) : preformula 0 -- notation ∀'

def formula := preformula L 0
```

First-order logic

To test our implementation, we formalized the completeness and compactness theorems.

```
theorem completeness {L : Language} (T : Theory L) (ψ :
    sentence L) : T ⊢' ψ ↔ T ⊨ ψ
```

```
theorem compactness {L : Language} {T : Theory L} {f : sentence
    L} :
  T ⊨ f ↔ ∃ fs : finset (sentence L), (↑fs : Theory L) ⊨ (f :
    sentence L) ∧ ↑fs ⊆ T
```

Generic extensions vs Boolean-valued models

Forcing goes something like this: given either a poset (of "forcing conditions") $\mathbb{P}$ or a Boolean completion $\mathbb{B}$ of $\mathbb{P}$, and a transitive ground model M of ZFC, one:

- Constructs a class of "names" ($\mathbb{P}$-names or $\mathbb{B}$-names)
- In the case of forcing with generic extensions, one selects a "generic filter" $G \subseteq \mathbb{P}$ and uses it to "evaluate" the $\mathbb{P}$-names, producing the forcing extension $M[G]$ which is checked to be a model of ZFC with the desired properties.
- In the case of Boolean-valued models, one works with the $\mathbb{B}$-names directly, as a $\mathbb{B}$-valued model $M^{\mathbb{B}}$-valued model of ZFC. This becomes the forcing extension.

Generic extensions vs Boolean-valued models

Major problem for a Lean user: everything is defined set-theoretically, and the set theory seems inextricable from the definition.

1 page into Kunen's chapter on forcing:

Definition 14.1. A set $F \subset P$ is a *filter* on P if

(14.1) (i) F is nonempty;
(ii) if $p \leq q$ and $p \in F$, then $q \in F$;
(iii) if $p, q \in F$, then there exists $r \in F$ such that $r \leq p$ and $r \leq q$.

A set of conditions $G \subset P$ is *generic* over M if

(14.2) (i) G is a filter on P;
(ii) if D is dense in P and $D \in M$, then $G \cap D \neq \emptyset$.

We also say that G is M-generic, or P-generic (over M), or just *generic*.

Generic extensions vs Boolean-valued models

At first glance, the situation is not much better for Boolean-valued models.

We now suppose given a complete Boolean algebra B, which we will assume to be fixed throughout the rest of this chapter. We also assume that B is a *set*, that is, $B \in V$.

We define the *universe* $V^{(B)}$ *of B-valued sets* by analogy with (1.2); namely, we define, by recursion on α,

$$V_\alpha^{(B)} = \{x: \mathrm{Fun}(x) \wedge \mathrm{ran}(x) \subseteq B \wedge \exists \xi < \alpha[\mathrm{dom}(x) \subseteq V_\xi^{(B)}]\} \tag{1.4}$$

and

$$V^{(B)} = \{x: \exists \alpha[x \in V_\alpha^{(B)}]\}. \tag{1.5}$$

Generic extensions vs Boolean-valued models

- Naiive approach: fix a model of ZFC in Lean, then replicate forcing arguments verbatim, *inside the model*. (Yikes).

- During formalization, do forcing arguments have to be carried out internally to a model of set theory?

- Answer: No!
- Use Boolean-valued approach to avoid generic filters.
- Key observation: the definition of $V^{\mathbb{B}}$ (equivalently, the name construction) is naturally implemented as an inductive type generalizing the Aczel construction of a model of ZFC from a universe of types.

A model of ZFC in Lean

The following construction is due to Aczel:

```
inductive pSet : Type (u+1)
| mk (α : Type u) (A : α → pSet) : pSet
```

- Note that `mk empty empty.elim` always exists, and corresponds to the empty set at the bottom of the von Neumann hierarchy.
- (Extensional) equivalence can be defined by structural recursion (the elimination principle for the inductive type `pSet` *is* $\in$-recursion): Two pre-sets are extensionally equivalent if every element of the first family is extensionally equivalent to some element of the second family and vice-versa.

The name construction done right

We add a third field to the constructor `pSet.mk`, so that all nodes of the tree are furthermore annotated with elements of $\mathbb{B}$ ("Boolean truth-values")

```
inductive bSet (𝔹 : Type u)
  [complete_boolean_algebra 𝔹] : Type (u+1)
| mk (α : Type u) (A : α → bSet) (B : α → 𝔹) : bSet
```

Note:

- When $\mathbb{B}$ is the singleton algebra `unit`, `bSet unit` is isomorphic to `pSet`.
- `bSet` $\mathbb{B}$ is exactly $V^{\mathbb{B}}$ (i.e. the name construction; `bSet` $\mathbb{B}$ comprises the "$\mathbb{B}$-names".)

The name construction

Compare with the set-theoretic definition of $\mathbb{P}$-names (Kunen):

2.5. DEFINITION. τ is a $\mathbb{P}$-name iff τ is a relation and

$$\forall \langle \sigma, p \rangle \in \tau \, [\sigma \text{ is a } \mathbb{P}\text{-name} \wedge p \in \mathbb{P}]. \quad \square$$

This definition does not mention models or any order on $\mathbb{P}$. The collection of $\mathbb{P}$-names will be a proper class if $\mathbb{P} \neq 0$.

Definition 2.5 must be understood as a definition by transfinite recursion. Formally, one defines the characteristic function of the $\mathbb{P}$-names, $\mathbf{H}(\mathbb{P}, \tau)$, by

$$\mathbf{H}(\mathbb{P}, \tau) = 1 \text{ iff } \tau \text{ is a relation} \wedge \forall \langle \sigma, p \rangle \in \tau \, [\mathbf{H}(\mathbb{P}, \sigma) = 1 \wedge p \in \mathbb{P}].$$

$$\mathbf{H}(\mathbb{P}, \tau) = 0 \text{ otherwise.}$$

Boolean-valued models of set theory

In `bSet` $\mathbb{B}$, ($\mathbb{B}$-valued) equality is defined by structural recursion:

```
def bv_eq : ∀ (x y : bSet), bool -- notation `=B`
| ⟨α,A,A′⟩ ⟨β,B,B′⟩ := ⨅ a : α, A′ a ⟹ ⨆ b : β, B′ b ⊓ bv_eq
    (A a) (B b) ⊓  ⨅ b : β, B′ b ⟹ ⨆ a : α, A′ a ⊓ bv_eq (A
    a) (B b)

def mem : bSet 𝔹 → bSet 𝔹 → 𝔹 --notation `∈B`
| a (mk α′ A′ B′) := ⨆a′, B′ a′ ⊓ a =B A′ a′
```

and ($\mathbb{B}$-valued) membership is defined from equality; together, these induce an assignment of truth-values (in $\mathbb{B}$) to all sentences in the language of ZFC.

Theorem. For every $\mathbb{B}$, `bSet` $\mathbb{B}$ is a **Boolean-valued model** of ZFC.

High-level overview

- The usual argument for the independence of CH goes like this:
  - Force $\neg$CH using the Cohen poset, producing a model where CH is false, so $\neg$CH is consistent with ZFC, i.e. CH is unprovable from ZFC.
  - Gödel showed that CH is true in the constructible universe L, so CH is consistent with ZFC, i.e. $\neg$CH is unprovable from ZFC.

- In our formalization, we:
  - Force $\neg$CH using Boolean-valued models, i.e. by using a Boolean completion $\mathbb{B}_{\text{cohen}}$ of the Cohen poset and verifying that $\neg$CH has truth-value $\top$ in `bSet 𝔹_cohen`.
  - Instead of constructing L, we also force CH via collapse forcing, again with Boolean-valued models, i.e. by verifying that the truth value of CH is $\top$ in `bSet 𝔹_collapse`.

High-level overview

- To do forcing, we must analyze combinatorial properties of $\mathbb{B}$ or a densely-embedded poset $\mathbb{P}$ presenting $\mathbb{B}$, and determine how these properties influence the set-theoretic behavior of `bSet` $\mathbb{B}$.
- This entails studying how the structure of $\mathbb{B}$ induces relationships between e.g. Lean's cardinals/ordinals (equivalence classes of types) with the internal cardinals/ordinals of `bSet` $\mathbb{B}$.
- Required development of elementary set theory (ordinals, etc) internal to `bSet` $\mathbb{B}$.
- Altogether, most technically involved part of the formalization.

Timeline of project

- June 2018: saw Freek's list
- September 2018: started project
- October 2018: Floris joins, first-order logic + soundness theorem
- November 2018: Completeness theorem
- February 2019: Definition of `bSet`
- March 2019: Cohen forcing and unprovability of CH
- June 2019: Start on collapse forcing
- August 2019: Finish collapse forcing and unprovability of $\neg$CH (except construction of $\aleph_1$),
- September 2019: Construct $\aleph_1$, finish independence of CH

Total time: 1 year, 4 days

Summary

- Was it as easy as I hoped? Eventually took 20,000 LOC and 1 year to complete, so maybe not.
- Our translation of the forcing argument into type theory shows that a ground model of set theory is not a prerequisite for forcing. Boolean-valued Aczel sets built out of a universe of types are enough.
- Challenges: many parts of textbook expositions did not have type-theoretic analogues, and the forcing argument for CH via Boolean-valued models is not well-documented.
- Formalization elucidated the proofs, and some parts were even discovered using Lean.
- Domain specific automation is useful; Lean makes it easy to write.

Summary

Thank you!

- flypitch.github.io
- https://www.github.com/flypitch/flypitch