A formalization of forcing and the consistency of the failure of the continuum hypothesis

Jesse Michael Han[1]
Department of Mathematics, University of Pittsburgh
https://www.pitt.edu/~jmh288
jessemichaelhan@gmail.com

Floris van Doorn
Department of Mathematics, University of Pittsburgh

Abstract

We describe a formalization of forcing using Boolean-valued models in the Lean 3 theorem prover, including the fundamental theorem of forcing and a deep embedding of first-order logic with a Boolean-valued soundness theorem. As an application of our framework, we specialize our construction to a Boolean completion of the Cohen poset and formally verify in the resulting model the failure of the continuum hypothesis.

2012 ACM Subject Classification

Keywords and phrases Interactive theorem proving, formal verification, set theory, forcing, independence, continuum hypothesis, Boolean-valued models

Digital Object Identifier 10.4230/LIPIcs.ITP.2019.23

Acknowledgements The authors would like to thank the members of the Pitt-CMU Lean group, particularly Simon Hudon, Jeremy Avigad, Mario Carneiro, and Tom Hales for their feedback and suggestions; we are also grateful to Dana Scott and John Bell for their advice and correspondence.

Introduction

The continuum hypothesis states that there are no sets strictly larger than the countable natural numbers and strictly smaller than the uncountable real numbers. It was introduced by Cantor in 1878 and was the very first problem on Hilbert's list of twenty-three outstanding problems in mathematics. Gödel proved in 1938 [**?**] that the continuum hypothesis was consistent with ZFC, and later conjectured that the continuum hypothesis was independent of ZFC, i.e. neither provable nor disprovable from the ZFC axioms. In 1963, Paul Cohen developed *forcing* [**?**], which allowed him to prove the consistency of the negation of the continuum hypothesis, and therefore complete the independence proof. For this work, which marked the beginning of modern set theory, he was awarded a Fields medal—the only one to ever be awarded for a work in mathematical logic.

The work we describe in this paper is part of the Flypitch project[2], which aims to formalize the independence of the continuum hypothesis. Our results mark a major milestone towards that goal.

Our formalization is written in the Lean 3 theorem prover. Lean is an interactive proof assistant under active development at Microsoft Research [**?**] [**?**]. It implements the Calculus of Inductive Constructions and has a similar metatheory to Coq, adding definitional proof irrelevance, quotient types, and a noncomputable choice principle. There is a well-known encoding of ZFC into dependent type theory with CIC, due to Aczel and Werner, which

[1] Corresponding author.
[2] https://github.com/flypitch/flypitch/

has been implemented in Lean's mathematical components library. The fact that Lean's metatheory is powerful enough to encode a model of ZFC already allows us to perform metatheoretic arguments about ZFC which were unavailable to e.g. Paulson [**?**], who went to extreme lengths to circumvent them inside Isabelle/ZF. While this was a formidable task, we content ourselves with treating ZFC as a mathematical object of study, and freely ignore the restrictions which it would impose as a foundation for the metatheory.

Indeed, our formalization makes as much use of the expressiveness of Lean's dependent type theory as possible, using constructions which are impossible or unwieldy to encode in HOL, much less ZF: Lean's ordinals and cardinals, which are defined as equivalence classes of well-ordered types, live one universe level up and play a crucial role in the forcing argument; the models of set theory we construct require as input entire universes of types; our encoding of first-order logic crucially uses parametrized inductive types to equate type-correctness with well-formedness, eliminating the need for separate well-formedness proofs.

Why Boolean-valued models? The method of forcing with Boolean-valued models was developed by Solovay and Scott (and independently, Vopěnka) in '65-'66 [**?**] [**?**] as a simplification of Cohen's method. Some of these simplifications were incorporated by Shoenfield [**?**] into a general theory of forcing using partial orders, and it is in this form that forcing is usually practiced. While both approaches have essentially the same mathematical content (see e.g. the discussion in Kunen [**?**] or Jech [**?**]), there are several reasons why we chose Boolean-valued models for our formalization:

- **Modularity.** The theory of forcing with Boolean-valued models cleanly splits into several components (a general theory of Boolean-valued semantics for first-order logic, a library for calculations inside complete Boolean algebras, the construction of Boolean-valued models of set theory, and the specifics of the forcing argument itself) which could be formalized in parallel and then recombined.
- **Directness.** For the purposes of an independence proof, the Boolean-valued soundness theorem eliminates the need to produce a two-valued model. This approach also bypasses any requirement for the reflection theorem/Löwenheim-Skolem theorems, Mostowski collapse, countable transitive models, or genericity considerations for filters.
- **Novelty and reusability.** As far as we were able to tell, the Boolean-valued approach to forcing has never been formalized. Furthermore, while for the purposes of an independence proof, forcing with Boolean-valued models and forcing with countable transitive models accomplish the same thing, a general library for Boolean-valued semantics of a deeply embedded logic could be used for formal verification applications outside of set theory, e.g. to formalize the Boolean-valued semantics of the stochastic λ-calculus [**?**].
- **Amenability to structural induction.** As with Coq, Lean is able to encode extremely complex objects and reason about their specifications using inductive types. However, the user must be careful to choose the encoding so that properties they wish to reason about are accessible by structural induction, which is the most natural mode of reasoning in the proof assistant. After observing (1) that the Aczel-Werner encoding of ZFC as an inductive type is essentially a special case of the recursive *name* construction from forcing (c.f. Section 3), and (2) that the automatically-generated induction principle for that inductive type *is* $\in$-induction, it is easy to see that this encoding can be modified to produce a Boolean-valued model of set theory where, again, $\in$-induction comes for free.

The rest of the paper is organized as follows. In Section 1 we outline the method of Boolean-valued models and sketch the forcing argument. Section 2 discusses a deep embedding of first-order logic, including a proof system, soundness and completeness theorems, and

crucially Boolean-valued semantics and the Boolean-valued soundness theorem. Section 3 discusses our construction of Boolean-valued models of set theory, emphasizing the usefulness of being able to metaprogram custom tactics to simulate predicate calculus inside an arbitrary complete Boolean algebra. Section 4 describes the formalization of the forcing argument and the construction of a suitable Boolean algebra for forcing $\neg\mathsf{CH}$. Section 5 describes the formalization of the Δ-system lemma, a technical result in transfinite combinatorics which ensures the preservation of cardinal inequalities. We conclude with an indication of future work towards a complete formal proof of the independence of the continuum hypothesis.

1 Outline of the proof

ZFC is a collection of first-order sentences in the language of a single binary predicate relation $\{\in\}$, used to axiomatize set theory. The continuum hypothesis can be written in this fashion as a first-order sentence CH. A proof of CH is a finite list of deductions starting from ZFC and ending at CH. The soundness theorem says that provability implies satisfiability, i.e. if $\mathsf{ZFC} \vdash \mathsf{CH}$, then CH interpreted in any model of ZFC is true. Taking the contrapositive, we can demonstrate the *unprovability* (equivalently, the consistency of the negation) of CH by exhibiting a single model where CH is not true.

A model of a first-order theory T is in particular a way of assigning true or false in a coherent way to sentences in the language. Modulo provable equivalence, the sentences form a Boolean algebra and "coherent" means the assignment is a Boolean algebra homomorphism (so $\wedge$ becomes meet, $\vee$ becomes join, $\forall$ becomes an indexed infimum, etc.) into $\mathbf{2} = \{\mathsf{true}, \mathsf{false}\}$. The soundness theorem ensures that this homomorphism v sends a proof $\phi \vdash \psi$ to an inequality $v(\phi) \leqslant v(\psi)$. But $\mathbf{2}$ does not really play a special role in this scheme, and may be replaced by any complete Boolean algebra $\mathbb{B}$, where the top and bottom elements $\top, \bot$ take the place of true and false. It is straightforward to extend this analogy to a $\mathbb{B}$-valued semantics for first-order logic. In particular, the upgraded soundness theorem now says that for any such $\mathbb{B}$, if $\mathsf{ZFC} \vdash \mathsf{CH}$, then for any $\mathbb{B}$-valued structure $\mathbf{M}$ where all the axioms of ZFC have truth-value $\top$, CH does also; as before, to demonstrate the consistency of the negation of CH it suffices to find just one $\mathbb{B}$ and a single $\mathbb{B}$-valued model where CH is not "true". That is where forcing comes in.

Before proceeding, it is profitable to keep in mind the following analogy, described by Scott in [**?**]. A ready supply of complete Boolean algebras $\mathbb{B}$ is obtained by taking the measure algebras of a probability space and quotienting by the ideal of events of measure zero. Let $\mathbf{M}$ be a $\mathbb{B}$-valued structure. A unary $\mathbb{B}$-valued predicate ϕ on $\mathbf{M}$ therefore assigns an event to every element m of $\mathbf{M}$, whose measure we can think of as being the probability that $\phi(m)$ is true. Specializing to the language of set theory, we can attach to every $m : \mathbf{M}$ an "indicator function" $\lambda x, x \in m$ which assigns to every x a probability that it is actually a member of m. Thus, by virtue of extensionality, we may think of the elements of a $\mathbb{B}$-valued model of ZFC as being "random sets".

Sources

Our strategy for constructing a Boolean-valued model in which the continuum hypothesis fails is a synthesis of the proofs in the textbooks of Bell ([**?**], Chapter 2) and Manin ([**?**], Chapter 8). (The latter actually has a direct proof that if $\mathbb{B}$ has the CCC, then there exists no surjection from $\aleph_1$ onto $\mathcal{P}(\aleph_0)$, but we found Bell's approach of directly constructing $\aleph_2$-many distinct Cohen reals and inducing an injection $\aleph_2 \hookrightarrow \mathcal{P}(\aleph_0)$ more satisfying.) For

the Δ-system lemma, which is required for the countable chain condition, we followed Kunen [?], formalizing the exact statement of Theorem 1.6.

Viewing the formalization

The source code for our formalization is available at `https://github.com/flypitch/flypitch`. The forcing argument for the negation of CH is located in `forcing.lean`. In a Lean-aware editor such as Emacs, the user is encouraged start at the theorem `neg_CH` and jump backwards to trace the dependencies of the proof.

2 First-order logic

2.1 (Pre)terms, (pre)formulas

2.2 Soundness

2.3 Completeness

As part of our formalization of first-order logic, we completed a verification of the Gödel completeness theorem. Although our present development of forcing did not require it, we anticipate that it will required later to e.g. prove the downward Löwenheim-Skolem theorem to extract countable transitive models for forcing with generic extensions; also, like the soundness theorem, it serves as a proof-of-concept and a stress-test of our chosen encoding of first-order logic.

For our formalization, we chose the Henkin-style approach of constructing a canonical term model. We sketch the argument: observe that the primary obstruction to being able to form a model of an arbitrary consistent theory from the terms of the language modulo provable equivalence is the possible lack of existential witnesses (for example, in the extreme case where the language is relational, like ZFC, and there are no terms at all.) The Henkin construction addresses this obstruction by iteratively expanding the language with new constant symbols axiomatized to be witnesses for existential statements from the previous stages; in the limit, one obtains a language with "enough witnessing constants" such that if the original theory was consistent, the terms do form a model.

There are several immediate challenges that must be addressed. First, since our first-order languages are represented by types instead of sets, we cannot really modify them in-place with new constant symbols. Instead, at each step of the construction, we must construct an entirely new language, in which the previous one embeds as a subtype, and in the limit, we cannot take a union, but must rather compute a directed colimit of types. Thus we developed a library for working with directed colimits of types.

In the process of our formalization, we discovered a nontrivial hole in many presentations of the Henkin argument (TODO(floris) describe `reflect_prf`)

We remark that our formalization of the completeness theorem is as general as possible, making no assumptions on the cardinality of the language or the presence of a function symbols.

2.4 Boolean-valued semantics for first-order logic

Quote: A main point of our exposition is that the well-known algebraic characterizations of cHa's and cBa's exactly mimic the rules of deduction in the respective logics...

2.5 The Boolean-valued soundness theorem

A soundness theorem says that a proof tree may be replayed to produce an actual proof in the object of truth-values. When the object is truth-values is the type `Prop` of propositions, this says that a proof tree becomes interpreted as a proof. When the object of truth-values is a Boolean algebra, this says that the proof tree becomes an internal implication from the interpretation of the context to the interpretation of the conclusion.

We designed our datatype of proofs as an inductive type whose constructors are precisely the natural deduction rules naturally supported by Lean's Prop. As a result, the proofs of either soundness theorem becomes a straightforward structural induction.

One complication which arose in the Boolean-valued case was keeping track of congruence lemmas...

Note that Boolean-valued equality is not really an equivalence relation.

3 Constructing Boolean-valued models of set theory

Throughout this section, we fix a universe level u, a type $\mathbb{B}$ `: Type u` and an instance of a complete Boolean algebra structure on $\mathbb{B}$.

In set theory (see e.g. Jech [**?**] or Bell [**?**]), Boolean-valued models are obtained by imitating the construction of the von Neumann cumulative hierarchy via a transfinite recursion where iterations of the powerset operation (taking functions into $\mathbf{2} = \{\text{true}, \text{false}\}$) are replaced by iterations of the "$\mathbb{B}$-valued powerset operation" (taking functions into $\mathbb{B}$).

Since this construction by transfinite recursion does not easily translate into type theory, our construction of Boolean-valued models of set theory is instead a variation on a well-known encoding originally due to Aczel [**?**] [**?**] [**?**]. This encoding was adapted by Werner [**?**] to encode ZFC into Coq, whose metatheory is close to that of Lean. Werner's construction was re-implemented in Lean's `mathlib` by Carneiro as part of [**?**]. In this approach, one takes a universe of types `Type u` as the starting point and then imitates the cumulative hierarchy by constructing the inductive type

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

(Just as the empty set kicks off the construction of the cumulative hierarchy, the empty type admits an empty map into any type and so induces `∅ : pSet`.)

The Aczel-Werner encoding is closely related to the recursive definition of *names* in forcing, which are used to construct the generic extensions of ground models when forcing over a ctm.

▸ **Definition 1.** *Let P be a partial order (which one thinks of as a collection of forcing conditions). A* P-name *is a collection of pairs (y, p) where y is a P-name and $p : P$.*

If P consists of only one element, then a P-name is specified by essentially the same information as a member of the inductive type `pSet` above. Conversely, specializing P to an arbitrary complete Boolean algebra $\mathbb{B}$, we modify the definition of `pSet.mk` so that elements are recursively assigned 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
```

Thus `bSet` $\mathbb{B}$ is the type of $\mathbb{B}$-names, and will be the underlying type of our Boolean-valued model of set theory.

3.1 Boolean-valued equality and membership

In `pSet`, equivalence of sets is defined by structural recursion as follows: two sets x and y are equivalent if and only if for every $w \in x$, there exists a $w' \in y$ such that w is equivalent to w', and vice-versa. Analogously, by translating quantifiers and connectives into operations on $\mathbb{B}$, Boolean-valued equality is defined in the same way:

```
def bv_eq : ∀ (x y : bSet 𝔹), 𝔹
| ⟨α, A, B⟩ ⟨α', A', B'⟩ :=
              (⨅a : α, B a ⟹ ⨆a', B' a' ⊓ bv_eq (A a) (A' a')) ⊓
                (⨅a' : α', B' a' ⟹ ⨆a, B a ⊓ bv_eq (A a) (A' a'))
```

3.2 Mixtures and the maximum principle

The maximum principle is the Boolean-valued incarnation of the axiom of choice in the metatheory. It says that an existential quantification over the entire model may be instantiated without changing the truth-value.

3.3 Check-names

From the definitions of `pSet` and `bSet`, one immediately sees that there is a canonical map `check : pSet → bSet 𝔹`, defined recursively as:

```
def check : pSet → bSet 𝔹
| ⟨α,A⟩ := ⟨α, λ a, check (A a), λ a, ⊤⟩
```

That is, `check` takes a `pSet` and recursively attaches the Boolean truth-value $\top$ to all elements. We call members of the image of `check` *check-names*. These are also known as *canonical names*, as they are the canonical representation of standard two-valued sets inside a Boolean-valued model of set theory.

3.4 Transfinite induction

In set theory, it is common to prove propositions via induction on an ordinal-valued rank function. In fact, this is how V^{BB} is typically constructed, by induction on the rank of sets in an existing universe of sets V. In Lean, this style of argument does not come for as free as, say, structural induction principles like $\in$-induction, which by virtue of the construction of bSet BB, *is* the induction principle for that inductive type. However, an interface is available for well-founded recursion on well-founded relations, and a development of the theory of ordinals as equivalence-classes of well-ordered types is available in `mathlib`. There were two places in the present work where transfinite induction was unavoidable, namely in the construction of an antichain for the maximum principle, and the verification that the canonical embedding of ordinals into `pSet` is injective.

3.5 Automation and metaprogramming

A key feature of Lean is that it is its own metalanguage, allowing for seamless in-line definitions of custom tactics. This feature was an invaluable asset for our formalization

We were also interested in whether the presence of a proof assistant would change the "user experience" of working with Boolean-valued models, for one possible complaint about forcing with Boolean-valued models is that calculating truth values inside a complete Boolean

algebra is notationally cumbersome in practice. This was indeed our experience—until we adapted Lean's automation to perform much of the bookkeeping for us, to the point where the user is able to pretend, with absolute rigor, that they are simply writing proofs in predicate calculus while lattice operations are being performed under the hood.

3.5.1 A custom proof language for reasoning in a Boolean-valued predicate calculus

3.5.2 Automating congruence lemmas with the simplifier

The simplifier is one of Lean 3's most sophisticated tools for automation, and is quite powerful when used correctly. It uses automatically-generated congruence lemmas to navigate under binders and perform rewrites using lemmas marked with the `simp` attribute. Given the pivotal role which congruence lemmas have in `simp`'s functionality, we consider it fitting that `simp` allowed us to automate the proofs of the Boolean-valued congruence (sometimes confusingly called *extensionality*) lemmas which are required to speak of a Boolean-valued predicate on a Boolean-valued structure.

TODO

3.6 The fundamental theorem of forcing

The fundamental theorem of forcing for Boolean-valued models [**?**] states that for any complete Boolean algebra B, V^B is a Boolean-valued model of ZFC. Since, in type theory, a universe `Type u` takes the place of the standard universe V, an analogous statement in our setting is that for every complete Boolean algebra $\mathbb{B}$, `bSet` $\mathbb{B}$ is a Boolean-valued model of ZFC.

After the development of the custom proof language, much of the verification of the axioms is routine, as the user is able to pretend they are working in ordinary 2-valued logic. We describe some of the interesting aspects of V^{BB} which are illuminated by the verification of the axioms, and which will be relevant later when forcing the negation of the continuum hypothesis.

- The axiom of infinity. What V^{BB} thinks is ω is actually ωCHECK. This is an instance of the Δ-zero-absoluteness theorem.

- The powerset operation. The powerset in V^{BB} is constructed as follows. TODO.

4 Forcing ¬CH

4.1 The Cohen poset and the regular open algebra

4.2 Adding ω_2-many distinct Cohen reals

4.3 Preservation of cardinal inequalities

5 Transfinite combinatorics and the countable chain condition

5.1 The Δ-system lemma

5.2 The countable chain condition

6 Future work

6.1 Applications of Boolean-valued semantics

(maybe remove? fill this out if you have time.)

6.2 Proof by reflection

Combined with the usual completeness theorem, the Boolean-valued soundness theorem will allow us to prove statements about a structure of the form bSet BB as follows: if the statement ϕ is provable from ZF, then we may prove that ZF proves ϕ by applying the completeness theorem to reason inside an arbitrary model of ZF. This avoids the complications of trying to work directly inside Boolean-valued logic. Then, given a Boolean-valued $\mathrm{L}_{\mathrm{ZFC}}$-structure $\mathbf{M}$ which satisfies ZF, the Boolean-valued soundness theorem tells us that this proof may be replayed inside $\mathbf{M}$ so that ϕ has truth-value greater than the truth-values of ZF, and is therefore satisfied inside $\mathbf{M}$.

In this way, we can transport proofs of statements such as "Zorn's lemma is equivalent to the axiom of choice", which is provable from ZF, directly from the world of 2-valued models to the world of Boolean-valued models.

We also remark that while our use of `simp` lemmas to generate congruence certificates sufficed for the purposes of this work, the "real" proof that something like the subset predicate is $=^B$-extensional is a proof by reflection: one constructs a formula which reifies the predicate, and then applies the fact that one is in the deeply-embedded Boolean-valued structure to obtain the congruence lemma automatically. We also intend to automate this.

6.3 Forcing with generic models

Our method does not support iterated forcing. Our method starts with a universe of types and uses that to construct a model of set theory.

6.4 Towards a formal proof of the independence of the continuum hypothesis

This work was carried out as part of the Flypitch project, which aims to formalize the independence of the continuum hypothesis from ZFC, i.e. that CH and its negation are both unprovable from the ZFC axioms.

Future goals of the project include:

- Various formulations of the axioms of ZFC are equiconsistent, including the versions used in this paper

7 Conclusions and future work

- In order to complete this formalization, we had to develop several libraries, e.g. for dependently-typed vectors, significant extensions of the lattice and Boolean algebra library, the theory of product topological space and their bases, and extensions to the set theory and ordinal libraries.
- `pSet` was not essential. Rather, in our type-theoretic foundations, the construction of a Boolean-valued standard universe of set theory has equal footing with the construction of an ordinary standard universe of set theory. We see that for the purposes of working with V^{BB}, V is no longer a prerequisite, but merely a useful tool for organizing the check-names.
- We used several features of our type-theoretic foundations to our advantage. We constructed a standard universe of set theory structurally in such a way that many properties of the underlying universe of types are reflected inside the model of set theory, and such that we get the axiom of regularity (more precisely, the principle of epsilon-induction) for free as the automatically-generated induction principle for our inductive type.
- Lean is great, meteoric growth — remark on recentness of developments in `mathlib` which made this possible (acknowledge developments from other theorem-provers, including porting of libraries e.g. `lattice` from Isabelle).
- Evidence that formalized mathematics is ready "in the large"

8 References

- Moore's The method of forcing
- Halmos-Givant Textbook on boolean algebras
- Gunther Pagano et al forcing in Isabelle/ZF
- Paulson constructible universe and set theory in Isabelle/ZF
- Sets in Coq, Coq in Sets
- Sets in types, types in sets
- Aczel's encoding of ZFC inside type theory