# How hard is it compute a failure of the continuum hypothesis?

Julia Kameryn Williams

Bard College at Simon's Rock

Logic Across Mathematics
2025 AWM Research Symposium
17 May 2025


Joint work with Joel David Hamkins (Notre Dame) and Russell Miller (CUNY).

# Hey, you got set theory in my computable structure theory!
# Hey, you got computable structure theory in my set theory!

# An introduction to forcing: forcing the continuum hypothesis

We will force CH by adding an $\omega_1$-sequence $G$ of all reals.

- The poset $\mathrm{Add}(\omega_1, 1)$ consists of conditions $p$ which are possible approximations to $G$: functions $\alpha \to 2$ for countable $\alpha$. Stronger conditions are longer binary sequences, which give more information about $G$.
- $G$ will be an $\omega_1$-length binary sequence, with every real coded at some point.

# An introduction to forcing: forcing the continuum hypothesis

We will force CH by adding an $\omega_1$-sequence $G$ of all reals.

- The poset $\mathrm{Add}(\omega_1, 1)$ consists of conditions $p$ which are possible approximations to $G$: functions $\alpha \to 2$ for countable $\alpha$. Stronger conditions are longer binary sequences, which give more information about $G$.
- $G$ will be an $\omega_1$-length binary sequence, with every real coded at some point.
- Not just any branch through the tree $\mathrm{Add}(\omega_1, 1)$ will code every real.
- To ensure this, we insist $G$ be generic—it must meet every dense set of conditions.
- For any $x : \omega \to 2$ and any condition $p$ you can extend $p$ to code $x$.
- A closure property ensures no new reals were added.

# An introduction to forcing: forcing the continuum hypothesis

We will force CH by adding an $\omega_1$-sequence $G$ of all reals.

- The poset $\mathrm{Add}(\omega_1, 1)$ consists of conditions $p$ which are possible approximations to $G$: functions $\alpha \to 2$ for countable $\alpha$. Stronger conditions are longer binary sequences, which give more information about $G$.
- $G$ will be an $\omega_1$-length binary sequence, with every real coded at some point.
- Not just any branch through the tree $\mathrm{Add}(\omega_1, 1)$ will code every real.
- To ensure this, we insist $G$ be generic—it must meet every dense set of conditions.
- For any $x : \omega \to 2$ and any condition $p$ you can extend $p$ to code $x$.
- A closure property ensures no new reals were added.

Genericity is important for more than just coding every real.

- Genericity forces $G \notin \mathrm{V}$.
- Genericity will ensure that the forcing extension $\mathrm{V}[G]$ satisfies the axioms of ZFC.

# Building the forcing extension

Our $\omega_1$-sequence $G$ doesn't just code all reals, it also tells us how to build the entire forcing extension $\mathrm{V}[G]$.

- Recursively define names, which describe objects in the larger universe.
- $G$ says how to interpret names: $\dot{x}^G$ is the interpretation of $\dot{x}$.
- There are definable forcing relations $p \Vdash \varphi(\dot{x}, \ldots)$ which control the behavior of $\mathrm{V}[G]$:

$$\mathrm{V}[G] \models \varphi(\dot{x}^G, \ldots) \Leftrightarrow \exists p \in G \; p \Vdash \varphi(\dot{x}, \ldots)$$

- Can use the genericity of $G$ to check that the axioms of ZFC are preserved in $\mathrm{V}[G]$.

# An intuitive sketch of forcing, in general

- A forcing poset $\mathbb{P}$ consists of possible approximations to a new object $G$.
- This $G$ will be a generic filter $\subseteq \mathbb{P}$.
- Recursively defined $\mathbb{P}$-names describe objects in the extension.
- Forcing relations $p \Vdash \varphi(\dot{x}, \ldots)$ control the behavior of $\mathrm{V}[G]$.

Three main parts of forcing:

- Getting a generic $G$;
- Interpreting the names to build the forcing extension;
- Using the forcing relations to determine satisfaction in the forcing extension.

# An intuitive sketch of forcing, in general

- A forcing poset $\mathbb{P}$ consists of possible approximations to a new object $G$.
- This $G$ will be a generic filter $\subseteq \mathbb{P}$.
- Recursively defined $\mathbb{P}$-names describe objects in the extension.
- Forcing relations $p \Vdash \varphi(\dot{x}, \ldots)$ control the behavior of $\mathrm{V}[G]$.

Three main parts of forcing:

- Getting a generic $G$;
- Interpreting the names to build the forcing extension;
- Using the forcing relations to determine satisfaction in the forcing extension.

The art is choosing $\mathbb{P}$ to force what you want.

| | |
|---|---|
| CH | $\mathrm{Add}(\omega_1, 1)$ |
| ¬CH | $\mathrm{Add}(\omega, \omega_2)$ |
| Whitehead conj. | $\mathrm{Add}(\omega_1, 1)$ |
| ¬Whitehead conj. | an $\omega_2$-iteration of ccc forcings |
| Borel conj. | an $\omega_2$-iteration of posets of Laver trees |
| ¬SCH | Prikry forcing at a large cardinal |

# The answer to the titular question: obviously not computable

# The answer to the titular question: obviously not computable

- Any computable process takes place entirely in $\mathrm{V}$, so it's not possible to produce $G$.
- Indeed, computation is absolute, so anything we could do in $\mathrm{V}[G]$ must already be in the ground model.
- We're dealing with uncountable objects and transfinite recursion.

# The answer to the titular question: obviously not computable

- Any computable process takes place entirely in $\mathrm{V}$, so it's not possible to produce $G$.
- Indeed, computation is absolute, so anything we could do in $\mathrm{V}[G]$ must already be in the ground model.
- We're dealing with uncountable objects and transfinite recursion.

If you know about the boolean algebra approach to forcing, the same problems recur.

- Building a complete boolean algebra $\mathbb{B}$ from a poset $\mathbb{P}$ and building a boolean topos $\mathrm{V}^{\mathbb{B}}$ from $\mathbb{B}$ are both infinitary processes.

For the titular question to be nontrivial we must mean something else.

# The multiverse of set theory

A standard approach in the model theory of set theory is to look at countable models of set theory.

- Rich tools are available and there is a robust multiverse to study.
- The Rasiowa–Sikorski lemma implies generics and thus forcing extensions over countable models always exist.
- Can think of a countable model of set theory as $\omega$ equipped with a binary relation $\in^M$.
- This is also an appropriate setting for computable structure theory.

# The multiverse of set theory

A standard approach in the model theory of set theory is to look at countable models of set theory.

- Rich tools are available and there is a robust multiverse to study.
- The Rasiowa–Sikorski lemma implies generics and thus forcing extensions over countable models always exist.
- Can think of a countable model of set theory as $\omega$ equipped with a binary relation $\in^M$.
- This is also an appropriate setting for computable structure theory.

Can formulate questions.
Given $M = (\omega, \in^M)$ and a poset $\mathbb{P} \in M$:

- Can we compute a generic $G$?
- Can we compute a representation of the forcing extension $M[G]$?
- Can we compute the elementary diagram of $M[G]$?

# The multiverse of set theory

A standard approach in the model theory of set theory is to look at countable models of set theory.

- Rich tools are available and there is a robust multiverse to study.
- The Rasiowa–Sikorski lemma implies generics and thus forcing extensions over countable models always exist.
- Can think of a countable model of set theory as $\omega$ equipped with a binary relation $\in^M$.
- This is also an appropriate setting for computable structure theory.

Can formulate questions.
Given $M = (\omega, \in^M)$ and a poset $\mathbb{P} \in M$:

- Can we compute a generic $G$?
- Can we compute a representation of the forcing extension $M[G]$?
- Can we compute the elementary diagram of $M[G]$?

**Warning!** No model of set theory can be computable simpliciter; we can only ask about computability relative to an oracle.

# Computing a generic $G$

**Theorem (Hamkins–Miller–W.)**

*Given the atomic diagram of $M = (\omega, \in^M)$ and a poset $\mathbb{P} \in M$ you can compute a generic $G$ for $\mathbb{P}$, given parameters.*

# Computing a generic $G$

**Theorem (Hamkins–Miller–W.)**

*Given the atomic diagram of $M = (\omega, \in^M)$ and a poset $\mathbb{P} \in M$ you can compute a generic $G$ for $\mathbb{P}$, given parameters.*

- The atomic diagram is simply the relation $\in^M$.
- Literally, $\mathbb{P}$ is an integer, not a set of conditions. Its extension is $\mathbb{P}^{\in} = \{n \in \omega : n \in^M \mathbb{P}\}$, and by computing $G$ I mean as a subset of $\mathbb{P}^{\in}$.

# Computing a generic $G$

**Theorem (Hamkins–Miller–W.)**

*Given the atomic diagram of $M = (\omega, \in^M)$ and a poset $\mathbb{P} \in M$ you can compute a generic $G$ for $\mathbb{P}$, given parameters.*

- The atomic diagram is simply the relation $\in^M$.
- Literally, $\mathbb{P}$ is an integer, not a set of conditions. Its extension is $\mathbb{P}^\in = \{n \in \omega : n \in^M \mathbb{P}\}$, and by computing $G$ I mean as a subset of $\mathbb{P}^\in$.

*Proof:* The usual proof of the Rasiowa–Sikorski is effective. □

# Computing a generic $G$

**Theorem (Hamkins–Miller–W.)**

*Given the atomic diagram of $M = (\omega, \in^M)$ and a poset $\mathbb{P} \in M$ you can compute a generic $G$ for $\mathbb{P}$, given parameters.*

- The atomic diagram is simply the relation $\in^M$.
- Literally, $\mathbb{P}$ is an integer, not a set of conditions. Its extension is $\mathbb{P}^{\in} = \{n \in \omega : n \in^M \mathbb{P}\}$, and by computing $G$ I mean as a subset of $\mathbb{P}^{\in}$.

*Proof:* The usual proof of the Rasiowa–Sikorski is effective. □

A little more detail: Fix the integers which are $\mathbb{P}$, $\leq_{\mathbb{P}}$, $\not\leq_{\mathbb{P}}$, $\perp_{\mathbb{P}}$, $\mathcal{D}$ the collection of dense subsets of $\mathbb{P}$ and this gives you the data needed to carry out the construction.

# Computing a generic $G$

**Theorem (Hamkins–Miller–W.)**

*Given the atomic diagram of $M = (\omega, \in^M)$ and a poset $\mathbb{P} \in M$ you can compute a generic $G$ for $\mathbb{P}$, given parameters.*

- The atomic diagram is simply the relation $\in^M$.
- Literally, $\mathbb{P}$ is an integer, not a set of conditions. Its extension is $\mathbb{P}^{\in} = \{n \in \omega : n \in^M \mathbb{P}\}$, and by computing $G$ I mean as a subset of $\mathbb{P}^{\in}$.

*Proof:* The usual proof of the Rasiowa–Sikorski is effective. □

A little more detail: Fix the integers which are $\mathbb{P}$, $\leq_{\mathbb{P}}$, $\not\leq_{\mathbb{P}}$, $\perp_{\mathbb{P}}$, $\mathcal{D}$ the collection of dense subsets of $\mathbb{P}$ and this gives you the data needed to carry out the construction.

**Caution!** Because we need to fix these integers non-uniformity is introduced; an isomorphic copy of the model will need to use a Turing machine which looks at different integers.

# What can we compute from the atomic diagram?

**Theorem (Hamkins–Miller–W.)**

*Let $X$ be a subset of a model $M$ of set theory. TFAE:*

- *There is a single c.e. operator which takes the atomic diagram of a presentation of $M$ and outputs the copy of $X$ for that presentation. ($X$ is uniformly r.i.c.e. in the atomic diagram.)*
- *Membership $a \in X$ is witnessed by a finite pattern of $\in$ in the transitive closure of $a$, with the list of patterns c.e. in the atomic diagram.*

# What can we compute from the atomic diagram?

All of the following predicates are not uniformly r.i.c.e. in the atomic diagram.

- $x = \emptyset$
- $x \subseteq y$
- $x$ is an ordered pair
- $x$ is a function
- $x$ is an ordinal
- $x = \omega$

**Theorem (Hamkins–Miller–W.)**

*Let $X$ be a subset of a model $M$ of set theory. TFAE:*

- *There is a single c.e. operator which takes the atomic diagram of a presentation of $M$ and outputs the copy of $X$ for that presentation. ($X$ is uniformly r.i.c.e. in the atomic diagram.)*
- *Membership $a \in X$ is witnessed by a finite pattern of $\in$ in the transitive closure of $a$, with the list of patterns c.e. in the atomic diagram.*

# What can we compute from the atomic diagram?

All of the following predicates are not uniformly r.i.c.e. in the atomic diagram.

- $x = \emptyset$
- $x \subseteq y$
- $x$ is an ordered pair
- $x$ is a function
- $x$ is an ordinal
- $x = \omega$

They are all $\Delta_0$ in the Lévy hierarchy: they can be expressed using only bounded quantifiers $\exists x \in y$ and $\forall x \in y$. This is the correct notion of the basic data of a model of set theory.

**Theorem (Hamkins–Miller–W.)**

*Let $X$ be a subset of a model $M$ of set theory. TFAE:*

- *There is a single c.e. operator which takes the atomic diagram of a presentation of $M$ and outputs the copy of $X$ for that presentation. ($X$ is uniformly r.i.c.e. in the atomic diagram.)*
- *Membership $a \in X$ is witnessed by a finite pattern of $\in$ in the transitive closure of $a$, with the list of patterns c.e. in the atomic diagram.*

# Computing the forcing extension $M[G]$

**Theorem (Hamkins–Miller–W.)**

*Take the $\Delta_0$-diagram for $M = (\omega, \in^M)$ as an oracle fix a poset $\mathbb{P} \in M$. Then we can computably produce $G$ an $M$-generic for $\mathbb{P}$ and a copy of $M[G]$.*

More precisely, we can compute a relation $\in^G \subseteq \omega^2$ so that $M[G] \cong (\omega, \in M[^G])$ and we can compute the canonical embedding $M \hookrightarrow M[G]$.

# Computing the forcing extension $M[G]$

**Theorem (Hamkins–Miller–W.)**

*Take the $\Delta_0$-diagram for $M = (\omega, \in^M)$ as an oracle fix a poset $\mathbb{P} \in M$. Then we can computably produce $G$ an $M$-generic for $\mathbb{P}$ and a copy of $M[G]$.*

More precisely, we can compute a relation $\in^G \subseteq \omega^2$ so that $M[G] \cong (\omega, \in M[^G])$ and we can compute the canonical embedding $M \hookrightarrow M[G]$.

*Proof sketch:* Everything we need is $\Delta_1$ and hence computable in the $\Delta_0$ diagram. □

# Computing the elementary diagram

**Theorem (Hamkins–Miller–W.)**

*Suppose we have the elementary diagram of $M = (\omega, \in^M)$ as an oracle and $\mathbb{P} \in M$ is a poset. Then we can computably produce $G$ an $M$-generic for $\mathbb{P}$ and the elementary diagram of a copy of $M[G]$.*

This also goes level by level. From the $\Sigma_n$-diagram we can compute the $\Sigma_n$-diagram for a copy of the extension.

# Computing the elementary diagram

**Theorem (Hamkins–Miller–W.)**

*Suppose we have the elementary diagram of $M = (\omega, \in^M)$ as an oracle and $\mathbb{P} \in M$ is a poset. Then we can computably produce $G$ an $M$-generic for $\mathbb{P}$ and the elementary diagram of a copy of $M[G]$.*

This also goes level by level. From the $\Sigma_n$-diagram we can compute the $\Sigma_n$-diagram for a copy of the extension.

*Proof sketch:* The map $\varphi \mapsto$ "$p \Vdash \varphi$" sending a formula to the corresponding forcing relation is computable, and we use the forcing relations to compute the elementary diagram. □

# Forcing is a computable procedure

Forcing is a computable procedure, with the level of information given as an oracle determining what we can compute about the extension.

- Given the atomic diagram for $M = (\omega, \in^M)$ and a poset $\mathbb{P} \in M$ we can compute a generic $G$ for $\mathbb{P}$ (using parameters).
- Given the $\Delta_0$-diagram we can moreover compute a copy of the extension $M[G]$ and its $\Delta_0$-diagram.
- Given the $\Sigma_n$-diagram we can compute the $\Sigma_n$-diagram of the extension.
- Given the elementary diagram we can compute the elementary diagram of the extension.

# So about that non-uniformity

- The construction of $G$ proceeded by searching through the conditions in $\mathbb{P}$ and the dense subsets of $\mathbb{P}$.
- A different presentation of $M$ will give a different order for the search, and produce a different $G$.
- In general, there will be $2^{\aleph_0}$ many possible $G$'s, so the $M[G]$ can't all be the same.

# So about that non-uniformity

- The construction of $G$ proceeded by searching through the conditions in $\mathbb{P}$ and the dense subsets of $\mathbb{P}$.
- A different presentation of $M$ will give a different order for the search, and produce a different $G$.
- In general, there will be $2^{\aleph_0}$ many possible $G$'s, so the $M[G]$ can't all be the same.

Altogether this tells us there is a non-uniformity to the process.

Can we get uniformity by a different process?

# Making the notion of uniformity precise: functoriality

For a structure $M$ let $\mathrm{Iso}(M)$ denote the category of isomorphic copies of $M$, with isomorphisms as its morphisms.

- A process to interpret $N$ in $M$ gives a map $F : \mathrm{Iso}(M) \to \mathrm{Iso}(N)$.
- If $F$ preserves isomorphisms then it is a functor.
- So asking for a uniform procedure to construct $M[G]$ from $M$ amounts to asking for a functor $F : \mathrm{Iso}(M) \to \mathrm{Iso}(M[G])$.

# Making the notion of uniformity precise: functoriality

For a structure $M$ let $\mathrm{Iso}(M)$ denote the category of isomorphic copies of $M$, with isomorphisms as its morphisms.

- A process to interpret $N$ in $M$ gives a map $F : \mathrm{Iso}(M) \to \mathrm{Iso}(N)$.
- If $F$ preserves isomorphisms then it is a functor.
- So asking for a uniform procedure to construct $M[G]$ from $M$ amounts to asking for a functor $F : \mathrm{Iso}(M) \to \mathrm{Iso}(M[G])$.

As computable structure theorists we don't want just any functor.

- A functor $F$ is computable if there is a Turing functional $\Phi$ which given info about an isomorphism $M \to M^*$ as an oracle will compute an isomorphism $F(M) \to F(M^*)$.

# Making the notion of uniformity precise: functoriality

For a structure $M$ let $\mathrm{Iso}(M)$ denote the category of isomorphic copies of $M$, with isomorphisms as its morphisms.

- A process to interpret $N$ in $M$ gives a map $F : \mathrm{Iso}(M) \to \mathrm{Iso}(N)$.
- If $F$ preserves isomorphisms then it is a functor.
- So asking for a uniform procedure to construct $M[G]$ from $M$ amounts to asking for a functor $F : \mathrm{Iso}(M) \to \mathrm{Iso}(M[G])$.

As computable structure theorists we don't want just any functor.

- A functor $F$ is computable if there is a Turing functional $\Phi$ which given info about an isomorphism $M \to M^*$ as an oracle will compute an isomorphism $F(M) \to F(M^*)$.
- (HTMMM 2017) There is a computable functor $F : \mathrm{Iso}(M) \to \mathrm{Iso}(N)$ iff $N$ is effectively interpretable in $M$.
- (HTMM 2018) If $F : \mathrm{Iso}(M) \to \mathrm{Iso}(N)$ is Baire-measurable then there is an infinitary interpretation $\mathcal{I}$ of $N$ in $M$ so that $F$ is naturally isomorphic to $F_{\mathcal{I}}$.

# Forcing is not a functorial process

**Theorem (Hamkins–Miller–W.)**

*If* ZFC *is consistent there is* $M \models$ ZFC *so that there is no computable functor* $\mathrm{Iso}(M) \to \mathrm{Iso}(M[G])$.

# Forcing is not a functorial process

**Theorem (Hamkins–Miller–W.)**

*If* ZFC *is consistent there is* $M \models$ ZFC *so that there is no computable functor* $\mathrm{Iso}(M) \to \mathrm{Iso}(M[G])$.

Indeed (Schlicht + HMW), can rule out a Borel functor mapping models to forcing extensions, even if we weaken "isomorphic" to "elementarily equivalent".

# Forcing is sometimes a functorial process

**Theorem (Hamkins–Miller–W.)**

*If $M$ is a pointwise-definable model of set theory there is a computable functor $\mathrm{Iso}(M) \to \mathrm{Iso}(M[G])$, using the full diagram of $M$ as its info.*

**Observation**

*Assume $\mathrm{V} = \mathrm{L}$. Then there is a $\Delta^1_2$ functor mapping presentations of countable models of set theory to forcing extensions which preserves isomorphism.*

# Forcing is sometimes a functorial process

**Theorem (Hamkins–Miller–W.)**

*If $M$ is a pointwise-definable model of set theory there is a computable functor $\mathrm{Iso}(M) \to \mathrm{Iso}(M[G])$, using the full diagram of $M$ as its info.*

**Observation**

*Assume $\mathrm{V} = \mathrm{L}$. Then there is a $\Delta^1_2$ functor mapping presentations of countable models of set theory to forcing extensions which preserves isomorphism.*

**Question**

- *Can there be an analytic (co-analytic) functorial method of producing forcing extensions?*
- *Does projective determinacy rule out a projective functorial method for producing forcing extensions?*

# Is forcing a computable procedure?

Positive results

- Given a presentation of a model of set theory we can compute its forcing extension.
- For special models we can do this in a functorial way.

Negative results

- But this procedure is in general dependent upon the choice of presentation.

That is, the procedure is computable in the model of set theory equipped with an $\omega$-enumeration of its elements, not merely in the model itself.

# Thank you!


- Joel David Hamkins, Russell Miller, and Kameryn Julia Williams, "Forcing as a computational process", *to appear: Computability*.
  Preprint: arXiv:2007.00418 [math.LO].
- Matthew Harrison-Trainor, Alexander Melkinov, Russell Miller, and Antonio Montalbán, "Computable functors and effective interpretability", JSL 82.1 (2017).
- Matthew Harrison-Trainor, Russell Miller, and Antonio Montalbán, "Borel functors and infinitary interpretations", JSL 83.4 (2018).