Set-theoretic forcing as a computational process

Joel David Hamkins
O'Hara Professor of Philosophy and Mathematics
University of Notre Dame

Associate Faculty, Professor of Logic
University of Oxford

Midwest Computability Seminar, 2 May 2023, University of Chicago

This is joint work.

[HMW20] Joel David Hamkins, Russell Miller, and Kameryn J. Williams, “Forcing as a computational process,” arXiv:2007.00418, http://jdh.hamkins.org/forcing-as-a-computational-process.

Forcing

The method of forcing is ubiquitous within set theory.

Forcing

The method of forcing is ubiquitous within set theory.

With forcing, we can build diverse models of set theory, revealing the vast range of set-theoretic possibility.

Forcing

The method of forcing is ubiquitous within set theory.

With forcing, we can build diverse models of set theory, revealing the vast range of set-theoretic possibility.

We construct models with the continuum hypothesis, or without, or models with Suslin trees, or without, or models with a definable well-ordering of the universe, or without.

Forcing

The method of forcing is ubiquitous within set theory.

With forcing, we can build diverse models of set theory, revealing the vast range of set-theoretic possibility.

We construct models with the continuum hypothesis, or without, or models with Suslin trees, or without, or models with a definable well-ordering of the universe, or without.

The method has an algebraic character.

Forcing

The method of forcing is ubiquitous within set theory.

With forcing, we can build diverse models of set theory, revealing the vast range of set-theoretic possibility.

We construct models with the continuum hypothesis, or without, or models with Suslin trees, or without, or models with a definable well-ordering of the universe, or without.

The method has an algebraic character.

For any model of set theory M, we can construct forcing extensions $M[G]$, akin to a field extension, where everything is constructible from objects in the ground model M and the new generic object G.

Main Question

I proposed years ago that we should mount an analysis of forcing from the perspective of computable structure theory.

Main Question

I proposed years ago that we should mount an analysis of forcing from the perspective of computable structure theory.

To what extent is forcing a computably effective process?

Main Question

I proposed years ago that we should mount an analysis of forcing from the perspective of computable structure theory.

To what extent is forcing a computably effective process?

Main Question

Given an oracle for a countable model of set theory M, to what extent can we compute its various forcing extensions $M[G]$?

Main Question

I proposed years ago that we should mount an analysis of forcing from the perspective of computable structure theory.

To what extent is forcing a computably effective process?

Main Question

Given an oracle for a countable model of set theory M, to what extent can we compute its various forcing extensions $M[G]$?

The answer depends on exactly how we are given the model M.

Main conclusions

In a variety of senses we get an affirmative answer.

Main conclusions

In a variety of senses we get an affirmative answer.

- With an oracle for the atomic diagram of M and $\mathbb{P} \in M$, we can compute an M-generic filter $G \subseteq \mathbb{P}$.

Main conclusions

In a variety of senses we get an affirmative answer.

- With an oracle for the atomic diagram of M and $\mathbb{P} \in M$, we can compute an M-generic filter $G \subseteq \mathbb{P}$.
- From the Δ_0 diagram of M, we can uniformly compute the Δ_0 diagram of $M[G]$.

Main conclusions

In a variety of senses we get an affirmative answer.

- With an oracle for the atomic diagram of M and $\mathbb{P} \in M$, we can compute an M-generic filter $G \subseteq \mathbb{P}$.
- From the Δ_0 diagram of M, we can uniformly compute the Δ_0 diagram of $M[G]$.
- From the full elementary diagram of M, we can uniformly compute the elementary diagram of $M[G]$.

Main conclusions

In a variety of senses we get an affirmative answer.

- With an oracle for the atomic diagram of M and $\mathbb{P} \in M$, we can compute an M-generic filter $G \subseteq \mathbb{P}$.
- From the Δ_0 diagram of M, we can uniformly compute the Δ_0 diagram of $M[G]$.
- From the full elementary diagram of M, we can uniformly compute the elementary diagram of $M[G]$.

Nevertheless in other senses the answer is negative.

Main conclusions

In a variety of senses we get an affirmative answer.

- With an oracle for the atomic diagram of M and $\mathbb{P} \in M$, we can compute an M-generic filter $G \subseteq \mathbb{P}$.
- From the Δ_0 diagram of M, we can uniformly compute the Δ_0 diagram of $M[G]$.
- From the full elementary diagram of M, we can uniformly compute the elementary diagram of $M[G]$.

Nevertheless in other senses the answer is negative.

- There is no computable nor even Borel construction that is functorial, in that different presentations of M give rise to the same $M[G]$.

The atomic diagram knows very little

Suppose we are given model M by its atomic diagram.

The atomic diagram knows very little

Suppose we are given model M by its atomic diagram.

The atomic diagram of $\langle M, \in^M \rangle$ has all instances of $x \in^M y$ or $x \notin^M y$, written on oracle tape.

The atomic diagram knows very little

Suppose we are given model M by its atomic diagram.

The atomic diagram of $\langle M, \in^M \rangle$ has all instances of $x \in^M y$ or $x \notin^M y$, written on oracle tape.

It turns out, however, that very little is computable from this—we cannot identify even a single fixed element.

The atomic diagram knows very little

Suppose we are given model M by its atomic diagram.

The atomic diagram of $\langle M, \in^M \rangle$ has all instances of $x \in^M y$ or $x \notin^M y$, written on oracle tape.

It turns out, however, that very little is computable from this—we cannot identify even a single fixed element.

Theorem

For any countable model of set theory $\langle M, \in^M \rangle$ and any element $b \in M$, no algorithm will pick out the number representing b uniformly given an oracle for the atomic diagram of a copy of M.

The atomic diagram knows very little

Suppose we are given model M by its atomic diagram.

The atomic diagram of $\langle M, \in^M \rangle$ has all instances of $x \in^M y$ or $x \notin^M y$, written on oracle tape.

It turns out, however, that very little is computable from this—we cannot identify even a single fixed element.

Theorem

For any countable model of set theory $\langle M, \in^M \rangle$ and any element $b \in M$, no algorithm will pick out the number representing b uniformly given an oracle for the atomic diagram of a copy of M.

We cannot reliably find the empty set, the ordinal ω, nor $\mathbb{R}$, nor any particular set.

Let's give the proof.

Let's give the proof.

Theorem

For any countable model of set theory $\langle M, \in^M \rangle$ and any element $b \in M$, no algorithm will pick out the number representing b uniformly given an oracle for the atomic diagram of a copy of M.

Proof.

Let's give the proof.

Theorem

For any countable model of set theory $\langle M, \in^M \rangle$ and any element $b \in M$, no algorithm will pick out the number representing b uniformly given an oracle for the atomic diagram of a copy of M.

Proof.

Fix $b \in M$ and fix an oracle for the atomic diagram of a copy of M.

Let's give the proof.

Theorem

For any countable model of set theory $\langle M, \in^M \rangle$ and any element $b \in M$, no algorithm will pick out the number representing b uniformly given an oracle for the atomic diagram of a copy of M.

Proof.

Fix $b \in M$ and fix an oracle for the atomic diagram of a copy of M.

Suppose an algorithm has claimed to identify b.

Let's give the proof.

Theorem

For any countable model of set theory $\langle M, \in^M \rangle$ and any element $b \in M$, no algorithm will pick out the number representing b uniformly given an oracle for the atomic diagram of a copy of M.

Proof.

Fix $b \in M$ and fix an oracle for the atomic diagram of a copy of M.

Suppose an algorithm has claimed to identify b.

Only finitely much of the atomic diagram was inspected.

Let's give the proof.

Theorem

For any countable model of set theory $\langle M, \in^M \rangle$ and any element $b \in M$, no algorithm will pick out the number representing b uniformly given an oracle for the atomic diagram of a copy of M.

Proof.

Fix $b \in M$ and fix an oracle for the atomic diagram of a copy of M.

Suppose an algorithm has claimed to identify b.

Only finitely much of the atomic diagram was inspected.

We may find an alternative presentation M' of M, whose atomic diagram agrees on that part, but disagrees on which number represents b.

Let's give the proof.

Theorem

For any countable model of set theory $\langle M, \in^M \rangle$ and any element $b \in M$, no algorithm will pick out the number representing b uniformly given an oracle for the atomic diagram of a copy of M.

Proof.

Fix $b \in M$ and fix an oracle for the atomic diagram of a copy of M.

Suppose an algorithm has claimed to identify b.

Only finitely much of the atomic diagram was inspected.

We may find an alternative presentation M' of M, whose atomic diagram agrees on that part, but disagrees on which number represents b.

So the algorithm will get the wrong answer on M'. $\square$

Forcing

Consider a model of set theory $M \models \mathrm{ZFC}$.

Forcing

Consider a model of set theory $M \models \mathrm{ZFC}$.

A forcing notion is a partial order $\mathbb{P} \in M$.

Forcing

Consider a model of set theory $M \models \mathrm{ZFC}$.

A forcing notion is a partial order $\mathbb{P} \in M$.

Conditions are $p, q \in \mathbb{P}$. Stronger conditions are lower $p \leq q$.

Forcing

Consider a model of set theory $M \models \mathrm{ZFC}$.

A forcing notion is a partial order $\mathbb{P} \in M$.

Conditions are $p, q \in \mathbb{P}$. Stronger conditions are lower $p \leq q$.

A filter $G \subseteq \mathbb{P}$ is *M-generic*, if for every dense set $D \subseteq \mathbb{P}$ in M, there is $p \in D \cap G$.

Forcing

Consider a model of set theory $M \models \mathrm{ZFC}$.

A forcing notion is a partial order $\mathbb{P} \in M$.

Conditions are $p, q \in \mathbb{P}$. Stronger conditions are lower $p \leq q$.

A filter $G \subseteq \mathbb{P}$ is *M-generic*, if for every dense set $D \subseteq \mathbb{P}$ in M, there is $p \in D \cap G$.

The forcing extension $M[G]$ will be built from $\mathbb{P}$-names $\tau \in M^{\mathbb{P}}$ and G.

How to build a generic filter

Consider a countable model $M \models \mathrm{ZFC}$.

How to build a generic filter

Consider a countable model $M \models \mathrm{ZFC}$.

Enumerate all dense subsets of $\mathbb{P}$ in M

$$D_0, D_1, D_2, \ldots$$

How to build a generic filter

Consider a countable model $M \models \mathrm{ZFC}$.

Enumerate all dense subsets of $\mathbb{P}$ in M

$$D_0, D_1, D_2, \ldots$$

Build a descending sequence of conditions

$$p_0 \geq p_1 \geq p_2 \geq p_3 \geq \cdots$$

How to build a generic filter

Consider a countable model $M \models \mathrm{ZFC}$.

Enumerate all dense subsets of $\mathbb{P}$ in M

$$D_0, D_1, D_2, \ldots$$

Build a descending sequence of conditions

$$p_0 \geq p_1 \geq p_2 \geq p_3 \geq \cdots$$

Start with $p_0 \in D_0$, and at stage n ensure $p_n \in D_n$.

How to build a generic filter

Consider a countable model $M \models \mathrm{ZFC}$.

Enumerate all dense subsets of $\mathbb{P}$ in M

$$D_0, D_1, D_2, \ldots$$

Build a descending sequence of conditions

$$p_0 \geq p_1 \geq p_2 \geq p_3 \geq \cdots$$

Start with $p_0 \in D_0$, and at stage n ensure $p_n \in D_n$.

This is possible, because each D_n is dense.

How to build a generic filter

Consider a countable model $M \models \mathrm{ZFC}$.

Enumerate all dense subsets of $\mathbb{P}$ in M

$$D_0, D_1, D_2, \ldots$$

Build a descending sequence of conditions

$$p_0 \geq p_1 \geq p_2 \geq p_3 \geq \cdots$$

Start with $p_0 \in D_0$, and at stage n ensure $p_n \in D_n$.

This is possible, because each D_n is dense.

The filter G generated by these conditions p_n is M-generic.

Construct G from Δ_0 diagram

We can carry out that construction using a oracle for the Δ_0 diagram of M (in set theory sense).

Construct G from Δ_0 diagram

We can carry out that construction using a oracle for the Δ_0 diagram of M (in set theory sense).

The oracle in effect provides a listing of M.

Construct G from Δ_0 diagram

We can carry out that construction using a oracle for the Δ_0 diagram of M (in set theory sense).

The oracle in effect provides a listing of M.

To begin, ask the oracle if there is p_0 in the set represented by 0 and $\mathbb{P}$. If so, go find one.

Construct G from Δ_0 diagram

We can carry out that construction using a oracle for the Δ_0 diagram of M (in set theory sense).

The oracle in effect provides a listing of M.

To begin, ask the oracle if there is p_0 in the set represented by 0 and $\mathbb{P}$. If so, go find one.

At stage n, ask if there is p_n below the previous in nth set and $\mathbb{P}$. If so, go find one.

Construct G from Δ_0 diagram

We can carry out that construction using a oracle for the Δ_0 diagram of M (in set theory sense).

The oracle in effect provides a listing of M.

To begin, ask the oracle if there is p_0 in the set represented by 0 and $\mathbb{P}$. If so, go find one.

At stage n, ask if there is p_n below the previous in nth set and $\mathbb{P}$. If so, go find one.

Can compute the filter G generated by conditions p_n.

Construct G from Δ_0 diagram

We can carry out that construction using a oracle for the Δ_0 diagram of M (in set theory sense).

The oracle in effect provides a listing of M.

To begin, ask the oracle if there is p_0 in the set represented by 0 and $\mathbb{P}$. If so, go find one.

At stage n, ask if there is p_n below the previous in nth set and $\mathbb{P}$. If so, go find one.

Can compute the filter G generated by conditions p_n.

So G is computable from Δ_0-diagram of M.

Actually, the atomic diagram suffices

Theorem

Given an oracle for the atomic diagram of $\langle M, \in^M \rangle \models \mathrm{ZF}$ *and* $\mathbb{P} \in M$*, we may compute an* M*-generic filter* $G \subseteq \mathbb{P}$*.*

Actually, the atomic diagram suffices

Theorem

Given an oracle for the atomic diagram of $\langle M, \in^M\rangle \models \mathrm{ZF}$ and $\mathbb{P} \in M$, we may compute an M-generic filter $G \subseteq \mathbb{P}$.

Proof.

Fix oracle for atomic diagram of $\langle M, \in^M\rangle$. Fix $\mathbb{P}$. Can decide $p \in \mathbb{P}$.

Actually, the atomic diagram suffices

Theorem

Given an oracle for the atomic diagram of $\langle M, \in^M\rangle \models \mathrm{ZF}$ and $\mathbb{P} \in M$, we may compute an M-generic filter $G \subseteq \mathbb{P}$.

Proof.

Fix oracle for atomic diagram of $\langle M, \in^M\rangle$. Fix $\mathbb{P}$. Can decide $p \in \mathbb{P}$.

Fix $\leq_{\mathbb{P}}$. Can recognize $p \leq q$. It's fussy—Kuratowski pairing $(p, q) = \{\{p\}, \{p, q\}\}$.

Actually, the atomic diagram suffices

Theorem

Given an oracle for the atomic diagram of $\langle M, \in^M \rangle \models \mathrm{ZF}$ and $\mathbb{P} \in M$, we may compute an M-generic filter $G \subseteq \mathbb{P}$.

Proof.

Fix oracle for atomic diagram of $\langle M, \in^M \rangle$. Fix $\mathbb{P}$. Can decide $p \in \mathbb{P}$.

Fix $\leq_{\mathbb{P}}$. Can recognize $p \leq q$. It's fussy—Kuratowski pairing $(p, q) = \{\{p\}, \{p, q\}\}$.

Fix set representing set of dense sets.

Actually, the atomic diagram suffices

Theorem

Given an oracle for the atomic diagram of $\langle M, \in^M \rangle \models \mathrm{ZF}$ and $\mathbb{P} \in M$, we may compute an M-generic filter $G \subseteq \mathbb{P}$.

Proof.

Fix oracle for atomic diagram of $\langle M, \in^M \rangle$. Fix $\mathbb{P}$. Can decide $p \in \mathbb{P}$.

Fix $\leq_{\mathbb{P}}$. Can recognize $p \leq q$. It's fussy—Kuratowski pairing $(p, q) = \{\{p\}, \{p, q\}\}$.

Fix set representing set of dense sets.

Can now build $p_0 \geq p_1 \geq p_2 \geq \cdots$, with p_n in the nth dense set.

Actually, the atomic diagram suffices

Theorem

Given an oracle for the atomic diagram of $\langle M, \in^M\rangle \models \mathrm{ZF}$ and $\mathbb{P} \in M$, we may compute an M-generic filter $G \subseteq \mathbb{P}$.

Proof.

Fix oracle for atomic diagram of $\langle M, \in^M\rangle$. Fix $\mathbb{P}$. Can decide $p \in \mathbb{P}$.

Fix $\leq_{\mathbb{P}}$. Can recognize $p \leq q$. It's fussy—Kuratowski pairing $(p, q) = \{\{p\}, \{p, q\}\}$.

Fix set representing set of dense sets.

Can now build $p_0 \geq p_1 \geq p_2 \geq \cdots$, with p_n in the nth dense set.

Can enumerate filter G generated by these p_n.

Actually, the atomic diagram suffices

Theorem

Given an oracle for the atomic diagram of $\langle M, \in^M \rangle \models \mathrm{ZF}$ and $\mathbb{P} \in M$, we may compute an M-generic filter $G \subseteq \mathbb{P}$.

Proof.

Fix oracle for atomic diagram of $\langle M, \in^M \rangle$. Fix $\mathbb{P}$. Can decide $p \in \mathbb{P}$.

Fix $\leq_{\mathbb{P}}$. Can recognize $p \leq q$. It's fussy—Kuratowski pairing $(p, q) = \{\,\{\,p\,\}, \{\,p, q\,\}\,\}$.

Fix set representing set of dense sets.

Can now build $p_0 \geq p_1 \geq p_2 \geq \cdots$, with p_n in the nth dense set.

Can enumerate filter G generated by these p_n.

Can actually decide G, not just enumerate it. Use $\perp_{\mathbb{P}}$. $\square$

Non-uniformity

The algorithm above is non-uniform in several senses.

Non-uniformity

The algorithm above is non-uniform in several senses.

- Required parameters for $\mathbb{P}$, $\leq_{\mathbb{P}}$, $\perp_{\mathbb{P}}$.
  (We can't compute these from atomic diagram.)

Non-uniformity

The algorithm above is non-uniform in several senses.

- Required parameters for $\mathbb{P}$, $\leq_{\mathbb{P}}$, $\perp_{\mathbb{P}}$.
  (We can't compute these from atomic diagram.)
- The particular G depends on the presentation.

Non-uniformity

The algorithm above is non-uniform in several senses.

- Required parameters for $\mathbb{P}$, $\leq_{\mathbb{P}}$, $\perp_{\mathbb{P}}$.
  (We can't compute these from atomic diagram.)
- The particular G depends on the presentation.
- The pairing function mattered.

Non-uniformity

The algorithm above is non-uniform in several senses.

- Required parameters for $\mathbb{P}$, $\leq_{\mathbb{P}}$, $\perp_{\mathbb{P}}$.
  (We can't compute these from atomic diagram.)
- The particular G depends on the presentation.
- The pairing function mattered.

In the last section, I shall consider whether a truly uniform construction is possible.

$\mathbb{P}$-names

A $\mathbb{P}$-name is a set consisting of pairs $\langle \sigma, p \rangle$, where σ a $\mathbb{P}$-name and $p \in \mathbb{P}$.

$\mathbb{P}$-names

A $\mathbb{P}$-name is a set consisting of pairs $\langle \sigma, p \rangle$, where σ a $\mathbb{P}$-name and $p \in \mathbb{P}$.

This is in effect a recursive definition on rank.

$\mathbb{P}$-names

A $\mathbb{P}$-name is a set consisting of pairs $\langle \sigma, p \rangle$, where σ a $\mathbb{P}$-name and $p \in \mathbb{P}$.

This is in effect a recursive definition on rank.

Lemma

The property of being a $\mathbb{P}$-name is Δ_1, hence decidable uniformly in the Δ_0-diagram of M.

$\mathbb{P}$-names

A $\mathbb{P}$-name is a set consisting of pairs $\langle \sigma, p \rangle$, where σ a $\mathbb{P}$-name and $p \in \mathbb{P}$.

This is in effect a recursive definition on rank.

Lemma

The property of being a $\mathbb{P}$-name is Δ_1, hence decidable uniformly in the Δ_0-diagram of M.

For any candidate σ, look for a transitive set that thinks it is a $\mathbb{P}$-name.

Building the forcing extension $M[G]$

How do we define the forcing extension $M[G]$?

Building the forcing extension $M[G]$

How do we define the forcing extension $M[G]$?

One standard approach is to define the “value” of each name

$$\operatorname{val}(\sigma, G) = \{ \operatorname{val}(\tau, G) \mid \exists p \in G \ \langle \tau, p \rangle \in \sigma \}.$$

Building the forcing extension $M[G]$

How do we define the forcing extension $M[G]$?

One standard approach is to define the "value" of each name

$$\mathrm{val}(\sigma, G) = \{\, \mathrm{val}(\tau, G) \mid \exists p \in G\ \langle \tau, p \rangle \in \sigma \,\}.$$

The forcing extension $M[G]$ consists of all such value sets.

Building the forcing extension $M[G]$

How do we define the forcing extension $M[G]$?

One standard approach is to define the "value" of each name

$$\mathrm{val}(\sigma, G) = \{ \mathrm{val}(\tau, G) \mid \exists p \in G \, \langle \tau, p \rangle \in \sigma \}.$$

The forcing extension $M[G]$ consists of all such value sets.

This approach, however, does not work with all models of set theory.

Building the forcing extension $M[G]$

How do we define the forcing extension $M[G]$?

One standard approach is to define the “value” of each name

$$\mathrm{val}(\sigma, G) = \{\, \mathrm{val}(\tau, G) \mid \exists p \in G\ \langle \tau, p\rangle \in \sigma \,\}.$$

The forcing extension $M[G]$ consists of all such value sets.

This approach, however, does not work with all models of set theory.

Because the recursion takes place outside M, it works only for well-founded models.

Building the forcing extension $M[G]$

How do we define the forcing extension $M[G]$?

One standard approach is to define the “value” of each name

$$\mathrm{val}(\sigma, G) = \{\, \mathrm{val}(\tau, G) \mid \exists p \in G\ \langle \tau, p \rangle \in \sigma \,\}.$$

The forcing extension $M[G]$ consists of all such value sets.

This approach, however, does not work with all models of set theory.

Because the recursion takes place outside M, it works only for well-founded models.

But we should want an understanding of forcing over *any* model of set theory.

Instead, construct $M[G]$ as Boolean ultrapower

Consider the forcing relations

$$p \Vdash_{\mathbb{P}} \sigma = \tau$$
$$p \Vdash_{\mathbb{P}} \sigma \in \tau$$

Instead, construct $M[G]$ as Boolean ultrapower

Consider the forcing relations

$$p \Vdash_{\mathbb{P}} \sigma = \tau$$
$$p \Vdash_{\mathbb{P}} \sigma \in \tau$$

For any ultrafilter $G \subseteq \mathbb{P}$, define

$$\sigma =_G \tau \quad \iff \quad \exists p \in G \quad p \Vdash_{\mathbb{P}} \sigma = \tau$$
$$\sigma \in_G \tau \quad \iff \quad \exists p \in G \quad p \Vdash_{\mathbb{P}} \sigma \in \tau.$$

The relation $=_G$ is a congruence with respect to the relation $\in_G$.

Instead, construct $M[G]$ as Boolean ultrapower

Consider the forcing relations

$$p \Vdash_{\mathbb{P}} \sigma = \tau$$
$$p \Vdash_{\mathbb{P}} \sigma \in \tau$$

For any ultrafilter $G \subseteq \mathbb{P}$, define

$$\begin{aligned} \sigma =_G \tau \quad &\iff \quad \exists p \in G \quad p \Vdash_{\mathbb{P}} \sigma = \tau \\ \sigma \in_G \tau \quad &\iff \quad \exists p \in G \quad p \Vdash_{\mathbb{P}} \sigma \in \tau. \end{aligned}$$

The relation $=_G$ is a congruence with respect to the relation $\in_G$.

Define the forcing extension $M[G]$ as quotient $M^{\mathbb{P}}/{=_G}$ using $\in_G$.

Instead, construct $M[G]$ as Boolean ultrapower

Consider the forcing relations

$$p \Vdash_{\mathbb{P}} \sigma = \tau$$
$$p \Vdash_{\mathbb{P}} \sigma \in \tau$$

For any ultrafilter $G \subseteq \mathbb{P}$, define

$$\sigma =_G \tau \quad \iff \quad \exists p \in G \quad p \Vdash_{\mathbb{P}} \sigma = \tau$$
$$\sigma \in_G \tau \quad \iff \quad \exists p \in G \quad p \Vdash_{\mathbb{P}} \sigma \in \tau.$$

The relation $=_G$ is a congruence with respect to the relation $\in_G$.

Define the forcing extension $M[G]$ as quotient $M^{\mathbb{P}}/=_G$ using $\in_G$.

The construction works with any ultrafilter—no need for genericity. But for G generic it agrees with the value construction.

Presenting $M[G]$

The presentation of $M[G]$ as the quotient of $M^{\mathbb{P}}$ by $=_G$ is computable from the Δ_0-diagram.

Presenting $M[G]$

The presentation of $M[G]$ as the quotient of $M^{\mathbb{P}}$ by $=_G$ is computable from the Δ_0-diagram.

Theorem

Given an oracle for Δ_0-diagram of $\langle M, \in^M \rangle \models \mathrm{ZF}$ and forcing $\mathbb{P} \in M$, we may compute an M-generic filter $G \subseteq \mathbb{P}$ and the Δ_0-elementary diagram of a presentation of $M[G]$.

Presenting $M[G]$

The presentation of $M[G]$ as the quotient of $M^{\mathbb{P}}$ by $=_G$ is computable from the Δ_0-diagram.

Theorem

Given an oracle for Δ_0-diagram of $\langle M, \in^M\rangle \models \mathrm{ZF}$ and forcing $\mathbb{P} \in M$, we may compute an M-generic filter $G \subseteq \mathbb{P}$ and the Δ_0-elementary diagram of a presentation of $M[G]$.

The diagram of the forcing extension $M[G]$ can be given in the full forcing language

$$\langle M[G], \in^{M[G]}, \check{M}, \sigma\rangle_{\sigma\in M^{\mathbb{P}}},$$

with a predicate $\check{M}$ for the ground model and constants for all the $\mathbb{P}$-names σ.

Presenting $M[G]$

The atomic forcing relations

$$p \Vdash_{\mathbb{P}} \sigma = \tau$$

$$p \Vdash_{\mathbb{P}} \sigma \in \tau,$$

have complexity Δ_1.

Presenting $M[G]$

The atomic forcing relations

$$p \Vdash_{\mathbb{P}} \sigma = \tau$$
$$p \Vdash_{\mathbb{P}} \sigma \in \tau,$$

have complexity Δ_1.

So given Δ_0 diagram of M and G, we can select representatives for $=_G$ classes.

Full forcing relation

For any formula φ,

$$M[G] \models \varphi[(\sigma_0)_G, \ldots, (\sigma_n)_G]$$

if and only if there is $p \in G$ so that

$$p \Vdash \varphi(\sigma_0, \ldots, \sigma_n).$$

Full forcing relation

For any formula φ,

$$M[G] \models \varphi[(\sigma_0)_G, \ldots, (\sigma_n)_G]$$

if and only if there is $p \in G$ so that

$$p \Vdash \varphi(\sigma_0, \ldots, \sigma_n).$$

The complexity of $p \Vdash \varphi$ is Σ_n if φ is (for $n \geq 1$).

Computing the full diagram

Theorem

Given an oracle for the full diagram of $\langle M, \in^M \rangle \models \mathrm{ZF}$ and forcing $\mathbb{P} \in M$, we can compute an M-generic filter and provide a computable presentation of full diagram of $M[G]$ in the forcing language.

Computing the full diagram

Theorem

Given an oracle for the full diagram of $\langle M, \in^M \rangle \models \mathrm{ZF}$ and forcing $\mathbb{P} \in M$, we can compute an M-generic filter and provide a computable presentation of full diagram of $M[G]$ in the forcing language.

Moreover, this goes level-by-level: given $\mathbb{P}$ and an oracle for the Σ_n-elementary diagram of M, for $n \geq 1$, we can decide the Σ_n-elementary diagram of this presentation.

Generic multiverse

The generic multiverse of a model of set theory M is the smallest collection of models closed under forcing extensions and grounds.

Grounds

W is a *ground* of M if $M = W[G]$ for some W-generic filter $G \subseteq \mathbb{P} \in W$.

Grounds

W is a *ground* of M if $M = W[G]$ for some W-generic filter $G \subseteq \mathbb{P} \in W$.

Ground model definability theorem (Laver, Woodin, JDH)

Every ground model is definable in its forcing extensions.

Grounds

W is a *ground* of M if $M = W[G]$ for some W-generic filter $G \subseteq \mathbb{P} \in W$.

Ground model definability theorem (Laver, Woodin, JDH)

Every ground model is definable in its forcing extensions.

The definition uses the approximation and cover properties.

Ground model enumeration theorem

The definability of grounds is uniform:

Ground model enumeration theorem

The definability of grounds is uniform:

Ground model enumeration theorem (Fuchs, Hamkins, Reitz)

There is an enumeration W_r of transitive classes such that

1. Every W_r is a ground of V.
2. Every ground of V is some W_r.
3. $x \in W_r$ is Π_2-definable $\varphi(r, x)$.

Ground model enumeration theorem

The definability of grounds is uniform:

Ground model enumeration theorem (Fuchs, Hamkins, Reitz)

There is an enumeration W_r of transitive classes such that

1. Every W_r is a ground of V.
2. Every ground of V is some W_r.
3. $x \in W_r$ is Π_2-definable $\varphi(r, x)$.

This theorem is the beginning of set-theoretic geology [FHR15].

Computing the grounds

Theorem

There is a uniform computable procedure which given an oracle for the Π_2-elementary diagram of a model of set theory $\langle M, \in^M \rangle \models \mathrm{ZFC}$ will compute a list of the Δ_0-diagrams of all the grounds of M.

Computing the grounds

Theorem

There is a uniform computable procedure which given an oracle for the Π_2-elementary diagram of a model of set theory $\langle M, \in^M \rangle \models \mathrm{ZFC}$ will compute a list of the Δ_0-diagrams of all the grounds of M.

From the Π_2-diagram, we get access to the ground-model enumeration W_r, and then we can compute what is true in them.

Computing the grounds

Theorem

Given an oracle for the full elementary diagram of a model of set theory $\langle M, \in^M\rangle \models \mathrm{ZFC}$, there is a computable procedure to compute a list of the full elementary diagrams of all the grounds of M.

Computing the grounds

Theorem

Given an oracle for the full elementary diagram of a model of set theory $\langle M, \in^M \rangle \models \mathrm{ZFC}$, there is a computable procedure to compute a list of the full elementary diagrams of all the grounds of M.

The elementary diagrams of the grounds of M can be computed from the elementary diagram of M, using the fact that the grounds themselves are uniformly definable.

Computing the generic multiverse

Question

Can we computably present the generic multiverse of M?

Computing the generic multiverse

Question

Can we computably present the generic multiverse of M?

Answer:

Computing the generic multiverse

Question

Can we computably present the generic multiverse of M?

Answer: No, because this is uncountable.

For example, for any given countable M, there are continuum many M-generic Cohen reals.

Computing the generic multiverse

Question

Can we computably present the generic multiverse of M?

Answer: No, because this is uncountable.

For example, for any given countable M, there are continuum many M-generic Cohen reals.

We shall seek a computable fragment: the *computable generic multiverse*.

Geology

Theorem (Usuba)

The grounds of a model of set theory are strongly downward directed.

Geology

Theorem (Usuba)

The grounds of a model of set theory are strongly downward directed.

Corollary (Fuchs,Hamkins,Reitz)

Every model in the generic multiverse of M is a forcing extension of a ground of M.

Geology

Theorem (Usuba)

The grounds of a model of set theory are strongly downward directed.

Corollary (Fuchs,Hamkins,Reitz)

Every model in the generic multiverse of M is a forcing extension of a ground of M.

That is, two steps suffice—go down to a ground, then up to a forcing extension.

Computable generic multiverse

Suppose we are given an oracle for the full elementary diagram of a model of set theory $\langle M, \in^M \rangle \models \mathrm{ZFC}$.

Computable generic multiverse

Suppose we are given an oracle for the full elementary diagram of a model of set theory $\langle M, \in^M \rangle \models \mathrm{ZFC}$.

We can uniformly compute the diagrams of all the various grounds $W_r \subseteq M$.

Computable generic multiverse

Suppose we are given an oracle for the full elementary diagram of a model of set theory $\langle M, \in^M \rangle \models \mathrm{ZFC}$.

We can uniformly compute the diagrams of all the various grounds $W_r \subseteq M$.

For any such ground W_r and any forcing notion $\mathbb{P} \in W_r$, any condition $p \in \mathbb{P}$, we can uniformly compute a W_r-generic filter $G \subseteq \mathbb{P}$ and the elementary diagram of $W_r[G]$.

Computable generic multiverse

Suppose we are given an oracle for the full elementary diagram of a model of set theory $\langle M, \in^M \rangle \models \mathrm{ZFC}$.

We can uniformly compute the diagrams of all the various grounds $W_r \subseteq M$.

For any such ground W_r and any forcing notion $\mathbb{P} \in W_r$, any condition $p \in \mathbb{P}$, we can uniformly compute a W_r-generic filter $G \subseteq \mathbb{P}$ and the elementary diagram of $W_r[G]$.

These models constitute a computable proxy for the generic multiverse of M.

Not capturing the full generic multiverse

Meanwhile, the computable generic multiverse is not dense in the full generic multiverse.

Not capturing the full generic multiverse

Meanwhile, the computable generic multiverse is not dense in the full generic multiverse.

For any countable model $M \models \mathrm{ZFC}$, we can find a forcing extension $M[G]$, such that no outer model of it has a Δ_0-diagram computable from the full elementary diagram of M.

Not capturing the full generic multiverse

Meanwhile, the computable generic multiverse is not dense in the full generic multiverse.

For any countable model $M \models \mathrm{ZFC}$, we can find a forcing extension $M[G]$, such that no outer model of it has a Δ_0-diagram computable from the full elementary diagram of M.

The method borrows techniques from non-amalgamation and pointwise definability results. We code a catastrophic real z into G in such a way, that any presentation of such a model will compute z.

Computable non-amalgamation

From the elementary diagram of a model $\langle M, \in^M \rangle \models \mathrm{ZFC}$, we can compute distinct M-generic Cohen reals c and d, such that $M[c]$ and $M[d]$ are non-amalgamable—they have no common extension to a model of ZFC with the same ordinals as M.

Computable non-amalgamation

From the elementary diagram of a model $\langle M, \in^M \rangle \models \mathrm{ZFC}$, we can compute distinct M-generic Cohen reals c and d, such that $M[c]$ and $M[d]$ are non-amalgamable—they have no common extension to a model of ZFC with the same ordinals as M.

In fact, we can make large assemblages with specified non-amalgamability.

Class forcing

Question

To what extent can we compute generics for class forcing?

Class forcing

Question

To what extent can we compute generics for class forcing?

It turns out that many of the arguments go through for class forcing.

Class forcing

Question

To what extent can we compute generics for class forcing?

It turns out that many of the arguments go through for class forcing.

Theorem

Given an oracle for the full elementary diagram of a countable model $\langle M, \in^M \rangle \models \mathrm{ZF}$ and given a definable pretame class forcing $\mathbb{P} \subseteq M$, there is a computable procedure to compute an M-generic filter $G \subseteq \mathbb{P}$ and decide the full elementary diagram of the forcing extension $M[G]$.

Second-order set theory

It is natural to undertake class forcing with second-order set theories, such as Gödel-Bernays set theory GBC.

Second-order set theory

It is natural to undertake class forcing with second-order set theories, such as Gödel-Bernays set theory GBC.

A model of GBC has both sets and classes.

Second-order set theory

It is natural to undertake class forcing with second-order set theories, such as Gödel-Bernays set theory GBC.

A model of GBC has both sets and classes.

Theorem

Let $\langle \mathcal{M}, \in^{\mathcal{M}} \rangle$ be a countable model of GB and suppose $\mathbb{P} \in \mathcal{M}$ is a class forcing notion with its atomic forcing relation a class in $\mathcal{M}$. Then, from an oracle for the Δ^1_0-elementary diagram of $\mathcal{M}$ there is a computable procedure to compute an $\mathcal{M}$-generic filter $G \subseteq \mathbb{P}$ and the Δ^1_0-diagram of $\mathcal{M}[G]$.

Second-order set theory

It is natural to undertake class forcing with second-order set theories, such as Gödel-Bernays set theory GBC.

A model of GBC has both sets and classes.

Theorem

Let $\langle\mathcal{M},\in^{\mathcal{M}}\rangle$ be a countable model of GB and suppose $\mathbb{P}\in\mathcal{M}$ is a class forcing notion with its atomic forcing relation a class in $\mathcal{M}$. Then, from an oracle for the Δ^1_0-elementary diagram of $\mathcal{M}$ there is a computable procedure to compute an $\mathcal{M}$-generic filter $G\subseteq\mathbb{P}$ and the Δ^1_0-diagram of $\mathcal{M}[G]$.

And a similar result holds for the full elementary diagrams.

Is the process functorial?

We have described how to compute a presentation of $M[G]$ from a presentation of M.

Is the process functorial?

We have described how to compute a presentation of $M[G]$ from a presentation of M.

Question

Is the procedure uniform in the sense that isomorphic presentations of M give rise to isomorphic presentations of $M[G]$?

Is the process functorial?

We have described how to compute a presentation of $M[G]$ from a presentation of M.

Question

Is the procedure uniform in the sense that isomorphic presentations of M give rise to isomorphic presentations of $M[G]$?

In other words, is the procedure well-defined with respect to isomorphisms of models?

Is the process functorial?

We have described how to compute a presentation of $M[G]$ from a presentation of M.

Question

Is the procedure uniform in the sense that isomorphic presentations of M give rise to isomorphic presentations of $M[G]$?

In other words, is the procedure well-defined with respect to isomorphisms of models?

This is connected with the existence of computable functors from the category of models of set theory up to isomorphism.

G depends on presentation

We have provided a computable procedure

$$(M, \in^M, \mathbb{P}) \mapsto G$$

to compute an M-generic filter G from the atomic diagram of M and forcing notion $\mathbb{P} \in M$.

G depends on presentation

We have provided a computable procedure

$$(M, \in^M, \mathbb{P}) \mapsto G$$

to compute an M-generic filter G from the atomic diagram of M and forcing notion $\mathbb{P} \in M$.

But the particular G we get can depend on how M is presented.

G depends on presentation

We have provided a computable procedure

$$(M, \in^M, \mathbb{P}) \mapsto G$$

to compute an M-generic filter G from the atomic diagram of M and forcing notion $\mathbb{P} \in M$.

But the particular G we get can depend on how M is presented.

The presentation led to a particular enumeration of the dense sets

$$D_0, D_1, D_2, \ldots$$

Changing this could affect G.

G depends on presentation

We have provided a computable procedure

$$(M, \in^M, \mathbb{P}) \mapsto G$$

to compute an M-generic filter G from the atomic diagram of M and forcing notion $\mathbb{P} \in M$.

But the particular G we get can depend on how M is presented.

The presentation led to a particular enumeration of the dense sets

$$D_0, D_1, D_2, \ldots$$

Changing this could affect G.

Can we avoid this sensitivity?

Inherently non-functorial

The answer is no: non-functoriality is inherent.

Inherently non-functorial

The answer is no: non-functoriality is inherent.

Theorem

There is no computable procedure taking the elementary diagram of a model $\langle M, \in^M\rangle \models \mathrm{ZFC}$ *with a partial order* $\mathbb{P}$ *to an* M*-generic filter* $G \subseteq \mathbb{P}$*, such that isomorphic copies of the input always give the corresponding isomorphic* G*.*

Inherently non-functorial

The answer is no: non-functoriality is inherent.

Theorem

There is no computable procedure taking the elementary diagram of a model $\langle M, \in^M\rangle \models \mathrm{ZFC}$ with a partial order $\mathbb{P}$ to an M-generic filter $G \subseteq \mathbb{P}$, such that isomorphic copies of the input always give the corresponding isomorphic G.

In other words, there is no computable procedure to produce generic filters that is functorial in the category of presentations of models of set theory under isomorphism.

Proof.

Assume toward contradiction that we have a computable procedure

$$\Phi : \Delta(M, \in^M, \mathbb{P}) \quad \mapsto \quad G$$

Assume functorial: isomorphic presentations of M get isomorphic G.

Proof.

Assume toward contradiction that we have a computable procedure

$$\Phi : \Delta(M, \in^M, \mathbb{P}) \quad \mapsto \quad G$$

Assume functorial: isomorphic presentations of M get isomorphic G.

Take a model $M \models \mathrm{ZFC}$ with $M_\kappa \prec M$ some κ.

Proof.

Assume toward contradiction that we have a computable procedure

$$\Phi : \Delta(M, \in^M, \mathbb{P}) \quad \mapsto \quad G$$

Assume functorial: isomorphic presentations of M get isomorphic G.

Take a model $M \models \mathrm{ZFC}$ with $M_\kappa \prec M$ some κ.

Fix any nontrivial forcing $\mathbb{P} \in M_\kappa$.

Proof.

Assume toward contradiction that we have a computable procedure

$$\Phi : \Delta(M, \in^M, \mathbb{P}) \quad \mapsto \quad G$$

Assume functorial: isomorphic presentations of M get isomorphic G.

Take a model $M \models \mathrm{ZFC}$ with $M_\kappa \prec M$ some κ.

Fix any nontrivial forcing $\mathbb{P} \in M_\kappa$.

Idea: run the computational process inside M, using finite fragments of diagram of M_κ.

Proof.

Assume toward contradiction that we have a computable procedure

$$\Phi : \Delta(M, \in^M, \mathbb{P}) \quad \mapsto \quad G$$

Assume functorial: isomorphic presentations of M get isomorphic G.

Take a model $M \models \mathrm{ZFC}$ with $M_\kappa \prec M$ some κ.

Fix any nontrivial forcing $\mathbb{P} \in M_\kappa$.

Idea: run the computational process inside M, using finite fragments of diagram of M_κ.

Any such finite fragment is part of a presentation of M. So all will give same G.

Proof.

Assume toward contradiction that we have a computable procedure

$$\Phi : \Delta(M, \in^M, \mathbb{P}) \quad \mapsto \quad G$$

Assume functorial: isomorphic presentations of M get isomorphic G.

Take a model $M \models \mathrm{ZFC}$ with $M_\kappa \prec M$ some κ.

Fix any nontrivial forcing $\mathbb{P} \in M_\kappa$.

Idea: run the computational process inside M, using finite fragments of diagram of M_κ.

Any such finite fragment is part of a presentation of M. So all will give same G.

Thus, G would exist inside M, contradiction. $\square$

But sometimes possible

Meanwhile, for *some* models M, we can compute G functorially from any presentation of M with its diagram.

But sometimes possible

Meanwhile, for *some* models M, we can compute G functorially from any presentation of M with its diagram.

Specifically, any pointwise definable model M will have this feature.

But sometimes possible

Meanwhile, for *some* models M, we can compute G functorially from any presentation of M with its diagram.

Specifically, any pointwise definable model M will have this feature.

M is pointwise definable, if every element of M is definable without parameters.

Functoriality for some models

Theorem

There is a computable functor Φ, in which Φ takes as input the elementary diagram of any pointwise definable model $\langle M, \in^M\rangle \models \mathrm{ZFC}$ and a forcing notion $\mathbb{P} \in M$ and returns an M-generic $G \subseteq \mathbb{P}$ and the elementary diagram of $M[G]$. That is, if $\langle M^*, \in^{M^*}\rangle$ and $\langle M^\dagger, \in^{M^\dagger}\rangle$ are two isomorphic presentations of M then $\Phi(M^*, \in^{M^*}, \mathbb{P}^*) \cong \Phi(M^\dagger, \in^{M^\dagger}, \mathbb{P}^\dagger)$.

Functoriality for some models

Theorem

There is a computable functor Φ*, in which* Φ *takes as input the elementary diagram of any pointwise definable model* $\langle M, \in^M\rangle \models \mathrm{ZFC}$ *and a forcing notion* $\mathbb{P} \in M$ *and returns an* M*-generic* $G \subseteq \mathbb{P}$ *and the elementary diagram of* $M[G]$*. That is, if* $\langle M^*, \in^{M^*}\rangle$ *and* $\langle M^\dagger, \in^{M^\dagger}\rangle$ *are two isomorphic presentations of* M *then* $\Phi(M^*, \in^{M^*}, \mathbb{P}^*) \cong \Phi(M^\dagger, \in^{M^\dagger}, \mathbb{P}^\dagger)$*.*

The point is that the elementary diagram of a pointwise definable model provides a canonical presentation of it.

Non-functorality for Borel processes

The non-functorality result extends to the Borel context.

Non-functorality for Borel processes

The non-functorality result extends to the Borel context.

Theorem

Suppose ZF *is consistent. Then there is no Borel function*

$$(M, \in^M, \mathbb{P}) \quad \mapsto \quad G$$

producing M*-generic filters in a functorial manner.*

Non-functorality for Borel processes

The non-functorality result extends to the Borel context.

Theorem

Suppose ZF *is consistent. Then there is no Borel function*

$$(M, \in^M, \mathbb{P}) \quad \mapsto \quad G$$

producing M-generic filters in a functorial manner.

Indeed, we cannot even get such a Borel function so that if $\langle M^*, \in^{M^*}, \mathbb{P}^* \rangle$ and $\langle M^\dagger, \in^{M^\dagger}, \mathbb{P}^\dagger \rangle$ are elementarily equivalent then so are $\langle M^*[G^*], \in^{M^*[G^*]} \rangle$ and $\langle M^\dagger[G^\dagger], \in^{M^\dagger[G^\dagger]} \rangle$.

Functorial at projective level

Meanwhile, if we go to the projective level, then there will (consistently) be a functorial process.

Functorial at projective level

Meanwhile, if we go to the projective level, then there will (consistently) be a functorial process.

For example, if $V = L$, then in L given any countable oracle code for a structure, we can find the L-least isomorphic copy in a projective way, specifically at the level Δ^1_2.

Functorial at projective level

Meanwhile, if we go to the projective level, then there will (consistently) be a functorial process.

For example, if $V = L$, then in L given any countable oracle code for a structure, we can find the L-least isomorphic copy in a projective way, specifically at the level Δ^1_2.

If we now build $M[G]$ using this copy, it will be constant on the isomorphism class of M, which is a very strong way of respecting isomorphism. Whether we can push this lower down in the projective hierarchy remains open.

Functorial at projective level

Meanwhile, if we go to the projective level, then there will (consistently) be a functorial process.

For example, if $V = L$, then in L given any countable oracle code for a structure, we can find the L-least isomorphic copy in a projective way, specifically at the level Δ^1_2.

If we now build $M[G]$ using this copy, it will be constant on the isomorphism class of M, which is a very strong way of respecting isomorphism. Whether we can push this lower down in the projective hierarchy remains open.

Question

Is there an analytic (or co-analytic) functorial method to produce generics for models of set theory?

References I

[FHR15] Gunter Fuchs, Joel David Hamkins, and Jonas Reitz. "Set-theoretic geology". *Annals of Pure and Applied Logic* 166.4 (2015), pp. 464–501. ISSN: 0168-0072. DOI: 10.1016/j.apal.2014.11.004. arXiv:1107.4776[math.LO]. http://jdh.hamkins.org/set-theoreticgeology.

[HMW20] Joel David Hamkins, Russell Miller, and Kameryn J. Williams. "Forcing as a computational process". *Mathematics arXiv* (2020). Under review. arXiv:2007.00418[math.LO]. http://jdh.hamkins.org/forcing-as-a-computational-process.

Thank you.

Slides and articles available on http://jdh.hamkins.org.

Joel David Hamkins
Professor of Logic, University of Oxford
Sir Peter Strawson Fellow, University College, Oxford