Justification by Confluence

Jason Zesheng Chen

University of California, Irvine

zeshengc@uci.edu

2024/04/22

Setting the tone

Philosophy of Mathematical Practice

- Pay attention to the math.
- Pay closer attention to the mathematician.

Setting the tone

Philosophy of Mathematical Practice

- Pay attention to the math.
- Pay closer attention to the mathematician.

Disclaimer

So, there will be little to no claims about "nature of the world/math" today (Sorry!)

What you'll be seeing a lot today

1. The work may seem more anthropological/sociological/documentarian than philosophical.

What you'll be seeing a lot today

1. The work may seem more anthropological/sociological/documentarian than philosophical.
2. The default mode of presentation:

What you'll be seeing a lot today

1. The work may seem more anthropological/sociological/documentarian than philosophical.
2. The default mode of presentation:
3. "Here's something lots of mathematicians have said. Let's see if they all mean the same!"

What you'll be seeing a lot today

1. The work may seem more anthropological/sociological/documentarian than philosophical.
2. The default mode of presentation:
3. "Here's something lots of mathematicians have said. Let's see if they all mean the same!"
4. "X said P in the context of computability; Y said Q in the context of some other area. They actually are saying the same thing."

What you'll be seeing a lot today

1. The work may seem more anthropological/sociological/documentarian than philosophical.
2. The default mode of presentation:
3. "Here's something lots of mathematicians have said. Let's see if they all mean the same!"
4. "X said P in the context of computability; Y said Q in the context of some other area. They actually are saying the same thing."
5. "X said P in the context of computability; Y also said P in the context of computability. They actually mean different things."

This is inspired by and modeled after...

Maddy (2019), *What do we want a foundation to do*

What does "foundation" mean? What do mathematicians mean when they say a theory provides a foundation for mathematics?

This is inspired by and modeled after...

Maddy (2019), *What do we want a foundation to do*

Conclusion: there is a wide array of foundational roles, orthogonal to each other.

This is inspired by and modeled after...

Maddy (2019), *What do we want a foundation to do*

Conclusion: there is a wide array of foundational roles, orthogonal to each other.

"It seems to me that the considerations the combatants offer against opponents and for their preferred candidates, as well as the roles each candidate actually or potentially succeeds in playing, reveal quite a number of different jobs that mathematicians want done."

This is inspired by and modeled after...

Maddy (2019), *What do we want a foundation to do*

Conclusion: there is a wide array of foundational roles, orthogonal to each other.

"It seems to me that the considerations the combatants offer against opponents and for their preferred candidates, as well as the roles each candidate actually or potentially succeeds in playing, reveal quite a number of different jobs that mathematicians want done."

Generous Arena, Shared Standards, Risk Assessment, Metamathematical Corral, Essential Guidance

My target of analysis

Justification by Confluence

The fact that we have all these equivalences or otherwise similar results constitutes evidence for...

The Namesake example

Gandy (1988). *The confluence of ideas in 1936*

Kleene's Argument by Confluence:

The Namesake example

Gandy (1988). *The confluence of ideas in 1936*

Kleene's Argument by Confluence:

> *Several other characterizations of a class of effectively calculable functions ... have turned out to be equivalent to general recursiveness ... The fact that several notions which differ widely lead to the same class of functions is a strong indication that this class is fundamental. (Kleene, 1952, p. 319)*

Common understanding of Kleene's argument

Confluence provides evidence for the Church-Turing Thesis: effectively feasible = Turing-computable.

Common understanding of Kleene's argument

Confluence provides evidence for the Church-Turing Thesis: effectively feasible = Turing-computable.

My Guiding Question

What does confluence justify? What does appealing to confluence achieve?

Spoiler

- **Conjecture Heuristic**
- **Rigor Assurance**
- **Coding Invariance**
- **Joint-carving**
- **Remarkable Coincidence**
- **Counterexample Resistance**

Conjecture Heuristic

An episode from the pre-history of Church-Turing Thesis:

- Church came up with the λ formalisms, but couldn't show that it computes all the basic arithmetic functions. (1930s)

Conjecture Heuristic

An episode from the pre-history of Church-Turing Thesis:

- Church came up with the λ formalisms, but couldn't show that it computes all the basic arithmetic functions. (1930s)
- Kleene finished the last piece of the puzzle (predecessor function). (1932)

Conjecture Heuristic

An episode from the pre-history of Church-Turing Thesis:

- Church came up with the λ formalisms, but couldn't show that it computes all the basic arithmetic functions. (1930s)
- Kleene finished the last piece of the puzzle (predecessor function). (1932)
- "we got the idea that this could represent all calculable functions ... for us the first idea that λ-definability was general was after ... dicovering that everything you thought of that you wanted to prove λ-definable, you could." (Kleene)

Conjecture Heuristic

An episode from the pre-history of Church-Turing Thesis:

- Church came up with the λ formalisms, but couldn't show that it computes all the basic arithmetic functions. (1930s)
- Kleene finished the last piece of the puzzle (predecessor function). (1932)
- "we got the idea that this could represent all calculable functions ... for us the first idea that λ-definability was general was after ... dicovering that everything you thought of that you wanted to prove λ-definable, you could." (Kleene)
- still pretty familiar, right?

Conjecture Heuristic

An episode from the pre-history of Church-Turing Thesis:

- Church came up with the λ formalisms, but couldn't show that it computes all the basic arithmetic functions. (1930s)
- Kleene finished the last piece of the puzzle (predecessor function). (1932)
- "we got the idea that this could represent all calculable functions ... for us the first idea that λ-definability was general was after ... dicovering that everything you thought of that you wanted to prove λ-definable, you could." (Kleene)
- But: Gödel regarded Church's proposal of identifying effective calculability with λ-definability as thoroughly unsatisfactory (November 1935, Church's letter to Kleene)

Conjecture Heuristic

An episode from the pre-history of Church-Turing Thesis:

- Church came up with the λ formalisms, but couldn't show that it computes all the basic arithmetic functions. (1930s)
- Kleene finished the last piece of the puzzle (predecessor function). (1932)
- "we got the idea that this could represent all calculable functions ... for us the first idea that λ-definability was general was after ... dicovering that everything you thought of that you wanted to prove λ-definable, you could." (Kleene)
- But: Gödel regarded Church's proposal of identifying effective calculability with λ-definability as thoroughly unsatisfactory (November 1935, Church's letter to Kleene)
- "If [Gödel] would propose any definition of effective calculability which seemed even partially satisfactory, [Church] would undertake to prove that it was included in lambda-definability." (ibid.)

1934 at the IAS

- Gödel proposed his own candidate for effective calculability: general recursiveness.

1934 at the IAS

- Gödel proposed his own candidate for effective calculability: general recursiveness.
- "Does this embrace all effectively calculable functions, and is it equivalent to λ-definability?"

1934 at the IAS

- Gödel proposed his own candidate for effective calculability: general recursiveness.
- "Does this embrace all effectively calculable functions, and is it equivalent to λ-definability?"
- Church proved the equivalence shortly after.

Conjecture Heuristic

Point:

Seeing just the mini-confluence, Church gained not only credence in the correctness of his definition, but also a *heuristic for conjecturing that other definitions will be equivalent to it.*

In classical computability

Post (1936). *Finite combinatory processes*

"The writer expects the present formulation to turn out to be logically equivalent to recursiveness in the sense of the Gödel-Church development."

In classical computability

Post (1936). *Finite combinatory processes*

"The writer expects the present formulation to turn out to be logically equivalent to recursiveness in the sense of the Gödel-Church development."

Rogers (1987). *Theory of recursive functions and effective computability*

"In fact, if certain general (and reasonable) formal criteria are laid down for what may constitute a [specification of what counts as computation], it is possible to show that the class of partial functions obtained is always a subclass of the maximal class of all partial recursive functions."

Conjecture Heuristic, elsewhere

Montalbán (2019) on Turing degrees of naturally occurring sets

"[The examples ordered under Turing-reduction] do seem to form a hierarchy. There are many more examples one can get from elsewhere in mathematics and many more from computability theory that are still very natural ... All the examples we know are ordered in a line. [*Proceeds to show that they are all iterates of Turing jumps*]"

Conjecture Heuristic, elsewhere

Montalbán (2019) on Turing degrees of naturally occurring sets

"[The examples ordered under Turing-reduction] do seem to form a hierarchy. There are many more examples one can get from elsewhere in mathematics and many more from computability theory that are still very natural ... All the examples we know are ordered in a line. [*Proceeds to show that they are all iterates of Turing jumps*]"

Martin's Conjecture

"The naturally occurring sets are linearly ordered by Turing reducibility and are iterates of Turing jumps."

Conjecture Heuristic, elsewhere

Aaronson on why he thinks $P \neq NP$

The strongest argument for $P \neq NP$ involves the thousands of problems that have been shown to be NP-complete, and the thousands of other problems that have been shown to be in P. If just one of these problems had turned out to be both NP-complete and in P, that would've immediately implied $P = NP$. Thus, we could argue, the hypothesis has had thousands of chances to be "falsified by observation."

Conjecture Heuristic, elsewhere

By now, tens of thousands of problems have been proved to be *NP*-complete. They range in character from theorem proving to graph coloring to airline scheduling to bin packing to protein folding to auction pricing to VLSI design to minimizing soap films to winning at Super Mario Bros. Meanwhile, another cluster of tens of thousands of problems has been proved to lie in P ... Those range from primality to matching to linear and semidefinite programming to edit distance to polynomial factoring to hundreds of approximation tasks ... To prove $P = NP$, it would suffice to find ... a single polynomial-time equivalence [between any of the *NP*-complete problems and any of the P problems] ... In half a century, this hasn't happened.

Conjecture Heuristic

Once a pattern occurs numerous times, from numerous sources, one is justified in conjecturing that this pattern will continue. As the Aaronson quote makes it clear, the usual tropes of prediction, inductive evidence, and empirical confirmation and fallibility are all at work here.

Conjecture Heuristic

Once a pattern occurs numerous times, from numerous sources, one is justified in conjecturing that this pattern will continue. As the Aaronson quote makes it clear, the usual tropes of prediction, inductive evidence, and empirical confirmation and fallibility are all at work here.

- Prototypical example 1: Church's conjecture that Gödel's formalism will be equivalent
- Prototypical example 2: Post's expectation that his formalism will be equivalent
- Aaronson's reasoning for conjecturing $P \neq NP$

Rigor Assurance

Nies (2012). *Computability and Randomness*

Many other formal definitions for the intuitive notion of a computable function were proposed. All turned out to be equivalent. This lends evidence to the Church-Turing thesis which states that any intuitively computable function is computable in the sense of [Turing machines]. More generally, each informally given algorithmic procedure can be implemented by a Turing program. We freely use this thesis in our proofs: we give a procedure informally and then take it for granted that a Turing program implementing it exists.

Rigor Assurance

Nies (2012). *Computability and Randomness*

Many other formal definitions for the intuitive notion of a computable function were proposed. All turned out to be equivalent. This lends evidence to the Church-Turing thesis which states that any intuitively computable function is computable in the sense of [Turing machines]. More generally, each informally given algorithmic procedure can be implemented by a Turing program. We freely use this thesis in our proofs: we give a procedure informally and then take it for granted that a Turing program implementing it exists.

This is known in the literature as "proof by Church's Thesis (or proof by the Church-Turing Thesis)"

Rigor Assurance

Rogers (1987). *Theory of recursive functions and effective computability*

Such methods ... permit us to avoid cumbersome detail and to isolate crucial mathematical ideas from a background of routine manipulation. We shall see that much profound mathematical substance can be discussed, proved, and communicated in this way ... Proofs which rely on informal methods have, in their favor, all the evidence accumulated in favor of Church's Thesis.

Outside of classical computability

for an admissible ordinal α and $A \subseteq \alpha$ the following are equivalent:

1. A is $\sum_1$-definable in L_α.
2. A is computably enumerable by Koepke's α-Turing machines.
3. A is semi-decidable by Koepke's α-register machines.

Outside of classical computability

Greenberg (2020). *Two applications of admissible computability*

In general, working in α-computability, with experience, we apply some kind of Church-Turing thesis to α-computable functions ... we eventually cease to write down precise Σ_1 formulas ... Instead, we develop an intuition as to what constitutes "legal" α-computable manipulations of α-finite objects (elements of L_α), and get a sense of the "time" that a process takes; if it takes fewer than α steps, then it "halts".

Outside of classical computability

Greenberg (2020). *Two applications of admissible computability*

In general, working in α-computability, with experience, we apply some kind of Church-Turing thesis to α-computable functions ... we eventually cease to write down precise Σ_1 formulas ... Instead, we develop an intuition as to what constitutes "legal" α-computable manipulations of α-finite objects (elements of L_α), and get a sense of the "time" that a process takes; if it takes fewer than α steps, then it "halts".

Notice

Greenberg is certainly not expecting the reader to develop an intuition for what is effective infinitarily (whatever that means). He is expecting the reader to develop an intuition for how to translate informal argument into formal ones.

One more example

Hamkins & Lewis, *Infinite Time Turing Machines*

"We will assume complete familiarity with the notions of Turing machines and ordinals and, in describing our algorithms, take the high road to avoid getting bogged down in Turing machine minutiae. We hope the readers will appreciate our saving them from reading what would otherwise resemble computer code." (Hamkins & Lewis, 2000)

Rigor Assurance

The fact that numerous formalizations of the same intuitive concept turn out to be equivalent provides a kind of assurance that intuitive, informal, natural language used in proofs can always be safely translated back to formal language, if one wishes.

Rigor Assurance

The fact that numerous formalizations of the same intuitive concept turn out to be equivalent provides a kind of assurance that intuitive, informal, natural language used in proofs can always be safely translated back to formal language, if one wishes.

- Prototypical example 1: Nies's and Rogers's anticipation of using informal description of algorithm in proofs.
- Greenberg's expectation that the reader can translate informal argument about infitary computation into formal ones.

San Mauro's tension

The Standard View towards proofs by Church's Thesis (according to San Mauro (2018))

Proof by Church's Thesis = proof is left to the reader.

San Mauro's tension

The Standard View towards proofs by Church's Thesis (according to San Mauro (2018))

Proof by Church's Thesis = proof is left to the reader.

i.e., exactly the kind of time-saving practice familiar to mathematicians from any other field. No additional significance.

San Mauro's tension

The Standard View towards proofs by Church's Thesis (according to San Mauro (2018))

Proof by Church's Thesis = proof is left to the reader.

San Mauro: taking this attitude misses out on a key aspect of what makes computability unique.

San Mauro's illustration: construction arguments in computability

- Many computability arguments begin with enumerating (say) the r.e. sets.

San Mauro's illustration: construction arguments in computability

- Many computability arguments begin with enumerating (say) the r.e. sets.
- Using this enumeration we construct many other objects of interest.

San Mauro's illustration: construction arguments in computability

- Many computability arguments begin with enumerating (say) the r.e. sets.
- Using this enumeration we construct many other objects of interest.
- Taking the Standard View on this kind of constructions entails that one still cares about the enumeration, despite not being bothered to write it down.

San Mauro's illustration: construction arguments in computability

- Many computability arguments begin with enumerating (say) the r.e. sets.
- Using this enumeration we construct many other objects of interest.
- Taking the Standard View on this kind of constructions entails that one still cares about the enumeration, despite not being bothered to write it down.

San Mauro's illustration: construction arguments in computability

- Many computability arguments begin with enumerating (say) the r.e. sets.
- Using this enumeration we construct many other objects of interest.
- Taking the Standard View on this kind of constructions entails that one still cares about the enumeration, despite not being bothered to write it down. So the proof is technically a proof schema, giving a recipe of construction for each enumeration.
- But what makes computability unique is that the properties of interest remain invariant under different enumerations. They are in some sense absolute properties of the objects, not a result of the coding.

San Mauro's illustration: construction arguments in computability

- Many computability arguments begin with enumerating (say) the r.e. sets.
- Using this enumeration we construct many other objects of interest.
- Taking the Standard View on this kind of constructions entails that one still cares about the enumeration, despite not being bothered to write it down. So the proof is technically a proof schema, giving a recipe of construction for each enumeration.
- But what makes computability unique is that the properties of interest remain invariant under different enumerations. They are in some sense absolute properties of the objects, not a result of the coding.
- This dis-entanglement with formalisms is what makes computability unique and what proponents of the Standard View are missing out on.

My diagnosis

There's no serious disagreement between San Mauro and proponents of the Standard View. The apparent disagreement stems from a conflation between:

My diagnosis

There's no serious disagreement between San Mauro and proponents of the Standard View. The apparent disagreement stems from a conflation between:

- Having faith that my reader can fill in the formal details. (**Rigor Assurance**)
- versus
- Not caring how they choose to fill in the formal details. Because it doesn't matter. (something else)

My diagnosis

There's no serious disagreement between San Mauro and proponents of the Standard View. The apparent disagreement stems from a conflation between:

- Having faith that my reader can fill in the formal details. (**Rigor Assurance**)
- versus
- Not caring how they choose to fill in the formal details. Because it doesn't matter. (something else)

Burgess's Indifferentism

"The general phenomenon of the indifference of working mathematicians to certain kinds of decisions that have to be made in any codification of mathematics ... two analysts who wish to collaborate do not need to check whether they were taught the same definition of 'real number'."

Coding Invariance

Immediately after Kleene's argument by confluence

"The notion of λ-definability has the variants λ-K-definability ... and λ-δ-definability ... also there is a parallel development, started by [Schönfinkel, Curry, and Rosser], which leads to a notion that we may call combinatory definability, proved equivalent to λ-definability by Rosser."

Coding Invariance

Immediately after Kleene's argument by confluence

"The notion of λ-definability has the variants λ-K-definability ... and λ-δ-definability ... also there is a parallel development, started by [Schönfinkel, Curry, and Rosser], which leads to a notion that we may call combinatory definability, proved equivalent to λ-definability by Rosser."

Notice

This is a different facet of confluence than the equivalence between different formalisms like Turing machines and λ-calculus. This is invariance under different codifications of the same formalism.

Definition

A set $X \subseteq \mathbb{N}$ is hyperarithmetical iff it satisfies any of the following equivalent definitions:

1. X is effectively Borel (e.g., it has a recursive Borel code).
2. X is (lightface) Δ^1_1 definable in second-order arithmetic.
3. X is computable from ω_1^{ck}-many times of iterated Turing jumps.
4. $X \in L_{\omega_1^{\mathrm{ck}}}$.

Moschovakis (2016). *Hyperarithmetical Sets*

"Codings are useful for expressing succinctly uniform properties of coded sets. ... It is clear that propositions ... which hold uniformly for a certain coding also hold uniformly for every equivalent coding."

Moschovakis (2016). *Hyperarithmetical Sets*

"Codings are useful for expressing succinctly uniform properties of coded sets. ... It is clear that propositions ... which hold uniformly for a certain coding also hold uniformly for every equivalent coding."

Moschovakis (2016). *Hyperarithmetical Sets*

"For a classical example, consider the coding of recursive partial functions specified by [Kleene's Normal Form Theorem]. Its precise definition depends on the choice of computation model that we use, Turing machines, systems of recursive equations or whatever [that is, each choice of model gives rise to a different enumeration of the recursive partial functions], but all these codings are equivalent and so uniform propositions about them are coding invariant."

Failure of Coding Invariance

1. One of the earliest definitions of quantum Turing machines allowed for arbitrary transition amplitudes. (Bernstein & Vazirani, 1993)

Failure of Coding Invariance

1. One of the earliest definitions of quantum Turing machines allowed for arbitrary transition amplitudes. (Bernstein & Vazirani, 1993)
2. The amplitudes, being real numbers, can code all sorts other information. This enabled Adleman et al. (1997) to prove that the class of sets decidable with bounded error in polynomial time has uncountable cardinality and contains sets of all Turing degrees.

Failure of Coding Invariance

1. One of the earliest definitions of quantum Turing machines allowed for arbitrary transition amplitudes. (Bernstein & Vazirani, 1993)
2. The amplitudes, being real numbers, can code all sorts other information. This enabled Adleman et al. (1997) to prove that the class of sets decidable with bounded error in polynomial time has uncountable cardinality and contains sets of all Turing degrees.
3. Subsequent journal version of Bernstein and Vazirani (1997) corrected the definition to only allow efficiently computable transition amplitudes.

Failure of Coding Invariance

1. One of the earliest definitions of quantum Turing machines allowed for arbitrary transition amplitudes. (Bernstein & Vazirani, 1993)
2. The amplitudes, being real numbers, can code all sorts other information. This enabled Adleman et al. (1997) to prove that the class of sets decidable with bounded error in polynomial time has uncountable cardinality and contains sets of all Turing degrees.
3. Subsequent journal version of Bernstein and Vazirani (1997) corrected the definition to only allow efficiently computable transition amplitudes.
4. "we need to constrain the entries allowed in the transition function ... Otherwise, it is possible to smuggle hard-to-compute quantities into the transition amplitudes [such as the solution to the halting problem]."

Failure of Coding Invariance

1. One of the earliest definitions of quantum Turing machines allowed for arbitrary transition amplitudes. (Bernstein & Vazirani, 1993)
2. The amplitudes, being real numbers, can code all sorts other information. This enabled Adleman et al. (1997) to prove that the class of sets decidable with bounded error in polynomial time has uncountable cardinality and contains sets of all Turing degrees.
3. Subsequent journal version of Bernstein and Vazirani (1997) corrected the definition to only allow efficiently computable transition amplitudes.
4. Careful choices were then made in the paper to ensure that, in terms of computability (i.e., what are computable simplciter), the resulting machines are equivalent to classical Turing machines.
5. The resulting definition of quantum Turing machines is now the canon.

Example from logic: iterated consistency

Define $T_0 := \mathrm{ZFC}$, $T_{2^n} := T_n + \mathrm{Con}(T_n)$, $T_{3 \cdot 5^e} = \mathrm{ZFC} \cup \bigcup T_{\Phi_e(n)}$. Now coding peculiarity follows:

Example from logic: iterated consistency

Define $T_0 := \mathrm{ZFC}$, $T_{2^n} := T_n + \mathrm{Con}(T_n)$, $T_{3\cdot 5^e} = \mathrm{ZFC} \cup \bigcup T_{\Phi_e(n)}$. Now coding peculiarity follows:

Theorem (Turing's completeness theorem)

For every true Π^0_1 sentence φ, there exists a notation in d in Kleene's $\mathcal{O}$ such that φ is provable in T_d.

Example from logic: iterated consistency

Define $T_0 := \mathrm{ZFC}$, $T_{2^n} := T_n + \mathrm{Con}(T_n)$, $T_{3\cdot 5^e} = \mathrm{ZFC} \cup \bigcup T_{\Phi_e(n)}$. Now coding peculiarity follows:

Theorem (Turing's completeness theorem)

For every true Π^0_1 sentence φ, there exists a notation in d in Kleene's $\mathcal{O}$ such that φ is provable in T_d.

Lesson

If we want to talk about iterated consistency statements, then the theories in the limit are susceptible to coding peculiarities.

A quick comparison

Rigor Assurance

Confluence $\Rightarrow$ not having to worry about informal language breaking a proof.

Coding Invariance

Confluence $\Rightarrow$ not having to worry that different codifications end up making the proofs talk about something else entirely.

A very illuminating example

Invariant descriptive set theory

The abstract study of how difficult classification problems are. (E.g., "If I know how to tell whether these two quantities are identical, then can I tell whether these two structures are isomorphic?")

A very illuminating example

Invariant descriptive set theory

The abstract study of how difficult classification problems are. (E.g., "If I know how to tell whether these two quantities are identical, then can I tell whether these two structures are isomorphic?")

A worry

In IDST, we are not really talking about the mathematical structures themselves, but talk about coded versions of them.

A very illuminating example

Invariant descriptive set theory

The abstract study of how difficult classification problems are. (E.g., "If I know how to tell whether these two quantities are identical, then can I tell whether these two structures are isomorphic?")

A worry

In IDST, we are not really talking about the mathematical structures themselves, but talk about coded versions of them. E.g., a finitely generated countable group is coded by a function $f : \mathbb{N}^2 \to \mathbb{N}$, telling us how the group operation behaves; or as a subgroup of some universal group (say the free group F_ω).

A very illuminating example

Gao's Thesis, (Gao, 2008, p. 328)

For any class $\mathcal{H}$ of mathematical structures, if (X_1, Ω_1) and (X_2, Ω_2) are two standard Borel spaces naturally coding elements of $\mathcal{H}$, then there exists a Borel map $f : X_1 \to X_2$ such that $f(x)$ and x are isomorphic as mathematical structures for every $x \in \mathcal{H}$.

A very illuminating example

Gao's Thesis, (Gao, 2008, p. 328)

"Any choice of coding/presentation of a structure is as good as any other."

A very illuminating example

Gao's Thesis, (Gao, 2008, p. 328)

"Any choice of coding/presentation of a structure is as good as any other."

This is classic Coding Invariance. Compare it with Gao's earlier appeal to proof by the Church-Turing Thesis:

A very illuminating example

Gao's Thesis, (Gao, 2008, p. 328)

"Any choice of coding/presentation of a structure is as good as any other."

This is classic Coding Invariance. Compare it with Gao's earlier appeal to proof by the Church-Turing Thesis:

Gao (2008, p. 24)

All formal definitions of computability have been shown to be equivalent ... we will not deal with the details of the above definition, but will rather adopt the Church-Turing Thesis ... Thus if a function is intuitively computable by an informal algorithm then by the Church-Turing Thesis we may conclude that it is formally computable without checking the details of the formal definitions.

Rigor Assurance

Confluence $\Rightarrow$ not having to worry about invalidity brought by translating from informal to formal language.

Coding Invariance

Confluence $\Rightarrow$ not having to worry pseudo-insights smuggled in by the choice of coding.

Joint-carving

Gödel, 1946 Princeton Bicentennial lecture

"With this concept [of Turing machines] one has succeeded in giving an absolute definition of an interesting epistemological notion, i.e., one not depending on the formalism chosen."

Joint-carving

Gödel, 1946 Princeton Bicentennial lecture

"With this concept [of Turing machines] one has succeeded in giving an absolute definition of an interesting epistemological notion, i.e., one not depending on the formalism chosen."

Turing succeeded in providing *grounding* to confluence

"[besides the equivalences,] there is also grounding: the idea that, among the class of conceptually distinct precisifications of the given (intuitive) concept, one of them stands out as being indubitably adequate - as being the right idea." (Kennedy, 2020, p. 42)

In the empirical sciences

Nozick's Presidential Address at the 94th Eastern APA

"That invariance is importantly connected to something's being an objective fact is suggested by the practice of physicists, who treat what is invariant under Lorentz transformations as more objective than what varies under these transformations. Dirac writes, 'The important things in the world appear as the invariants... of ... transformations.' "

In the empirical sciences

Nozick's Presidential Address at the 94th Eastern APA

"That invariance is importantly connected to something's being an objective fact is suggested by the practice of physicists, who treat what is invariant under Lorentz transformations as more objective than what varies under these transformations. Dirac writes, 'The important things in the world appear as the invariants... of ... transformations.' "

Point:

Confluence $\Rightarrow$ something objective is being touched upon.

Classic Example: Perrin

- Jean Baptiste Perrin (1870-1942). Winner of Nobel Physics Prize 1926

Classic Example: Perrin

- Jean Baptiste Perrin (1870-1942). Winner of Nobel Physics Prize 1926
- Found dozen of ways to measure Avogadro's number. All returned the same value.

Classic Example: Perrin

- Jean Baptiste Perrin (1870-1942). Winner of Nobel Physics Prize 1926
- Found dozen of ways to measure Avogadro's number. All returned the same value.
- Key point: this kind of confluence was available before Perrin, in 1860s (Maddy, 2007, p. 404).

Classic Example: Perrin

- Jean Baptiste Perrin (1870-1942). Winner of Nobel Physics Prize 1926
- Found dozen of ways to measure Avogadro's number. All returned the same value.
- Key point: this kind of confluence was available before Perrin, in 1860s (Maddy, 2007, p. 404).
- "As though we can tip the scale on behalf of the correctness of this value by providing it with 13 as opposed to only 10 independent derivations."

Classic Example: Perrin

- Jean Baptiste Perrin (1870-1942). Winner of Nobel Physics Prize 1926
- Found dozen of ways to measure Avogadro's number. All returned the same value.
- Key point: this kind of confluence was available before Perrin, in 1860s (Maddy, 2007, p. 404).
- "As though we can tip the scale on behalf of the correctness of this value by providing it with 13 as opposed to only 10 independent derivations."
- Beyond the equivalent results, Perrin performed experiments (vertical distribution experiments) allowing for detection of the atoms, and confirming some of Einstein's predictions in the process.

Classic Example: Perrin

- Jean Baptiste Perrin (1870-1942). Winner of Nobel Physics Prize 1926
- Found dozen of ways to measure Avogadro's number. All returned the same value.
- Key point: this kind of confluence was available before Perrin, in 1860s (Maddy, 2007, p. 404).
- "As though we can tip the scale on behalf of the correctness of this value by providing it with 13 as opposed to only 10 independent derivations."
- Beyond the equivalent results, Perrin performed experiments (vertical distribution experiments) allowing for detection of the atoms, and confirming some of Einstein's predictions in the process.
- This made the equivalents make sense. Much like Turing's analysis of computability made the confluence of definitions make sense for Gödel.

Joint-carving

Both Turing and Perrin "carved nature at its joints". They managed to identified a "joint of nature" that made clear what was it that other equivalents were converging upon.

A claim of no such "joint-detection"

Attempts to define "algorithmically random sequence" have seen a high degree of confluence: measure-theoretic, incompressibility no-winning-betting-strategy characterizations all turned out to be equivalent.

A claim of no such "joint-detection"

Attempts to define "algorithmically random sequence" have seen a high degree of confluence: measure-theoretic, incompressibility no-winning-betting-strategy characterizations all turned out to be equivalent. Does this suggest a "Church-Turing Thesis" for randomness?

Porter (2021). *The Equivalence of Definitions of Algorithmic Randomness*

There are no alternative definitions of computable function that are serious candidates for capturing the intuitive notion of computable function ... This uniqueness datum is not present in the theory of algorithmic randomness ... Appealing to equivalence results as evidence for the claim that a formal definition captures some informal notion is convincing only in the presence of a unique locus of equivalent definitions.

Porter (2021). *The Equivalence of Definitions of Algorithmic Randomness*

There are no alternative definitions of computable function that are serious candidates for capturing the intuitive notion of computable function ... This uniqueness datum is not present in the theory of algorithmic randomness ... Appealing to equivalence results as evidence for the claim that a formal definition captures some informal notion is convincing only in the presence of a unique locus of equivalent definitions.

No "grounding"

Having confluence alone doesn't help. The case of algorithmic randomness falters in the absence of a "joint" that the equivalents are converging upon.

Remarkable Coincidence

Feynman on the miracle of applied math

"There is a most remarkable coincidence: The equations for many different physical situations have exactly the same appearance." (Feynman et al., 1964).

Remarkable Coincidence

Feynman on the miracle of applied math

"There is a most remarkable coincidence: The equations for many different physical situations have exactly the same appearance." (Feynman et al., 1964).

Feynman and Maddy

Both agree that the "miracle" is not so miraculous, at least not pointing to some underlying mathematical structure of the world.

Remarkable Coincidence

Feynman on the miracle of applied math

"There is a most remarkable coincidence: The equations for many different physical situations have exactly the same appearance." (Feynman et al., 1964).

Feynman and Maddy

Both agree that the "miracle" is not so miraculous, at least not pointing to some underlying mathematical structure of the world.

But still...

"This means that having studied one subject, we immediately have a great deal of direct and precise knowledge about the solutions of the equations of another." (Feynman et al., 1964)

Although the confluence for algorithmic randomness fails to turn into a "Thesis", it is still a remarkable coincidence that all these definitions are equivalent.

Although the confluence for algorithmic randomness fails to turn into a "Thesis", it is still a remarkable coincidence that all these definitions are equivalent.

Carnielli & Epstein (2008). *Computability*

All the authors we have quoted above are in agreement on one thing: [Church's Thesis] is not part of mathematics, it is not part of the theory of recursive functions or Turing machines. Those theories are interesting in their own right even if Church's thesis were to be abandoned. Whatever else the Most Amazing Fact [the equivalence of definitions] establishes, it shows that the notion which is stable under so many different formulations must be fundamental.

Remarkable Coincidence

Remarkable Coincidence may be best thought of as Joint-carving suppresing any background notion of being correct or privileged: one need not be bothered with a pre-theoretic notion, or any criteria of naturalness or correctness.

Remarkable Coincidence

Remarkable Coincidence may be best thought of as Joint-carving suppresing any background notion of being correct or privileged: one need not be bothered with a pre-theoretic notion, or any criteria of naturalness or correctness.

Slogan

Joint-carving tells us something has been done correctly, Remarkable Coincidence tells us something can be done (or has been done) fruitfully.

Remarkable Coincidence: Rudolph's Thesis

The following theorem states that two main structures that might sensibly be said to model measure-preserving transformations have the same generic dynamical properties.

Theorem (Glasner and King, Rudolph)

*Let P be a dynamical property that has the property of Baire. Then P is generic in **MPT** iff P is generic in SIM$([0,1)^{\mathbb{Z}})$.*

Remarkable Coincidence: Rudolph's Thesis

The following theorem states that two main structures that might sensibly be said to model measure-preserving transformations have the same generic dynamical properties.

Theorem (Glasner and King, Rudolph)

*Let P be a dynamical property that has the property of Baire. Then P is generic in **MPT** iff P is generic in SIM$([0,1)^{\mathbb{Z}})$.*

Rudolph's Thesis (Foreman, 2010)

Any two models for the measure preserving transformations are equivalent and have the same generic dynamical properties.

Rudolph's Thesis (Foreman, 2010)

Any two models for the measure preserving transformations are equivalent and have the same generic dynamical properties.

Rudolph's Thesis (Foreman, 2010)

Any two models for the measure preserving transformations are equivalent and have the same generic dynamical properties.

In the same paper, Foreman proceeds to carefully and formally define what a *model* is and what *equivalent* means.

Rudolph's Thesis (Foreman, 2010)

Any two models for the measure preserving transformations are equivalent and have the same generic dynamical properties.

In the same paper, Foreman proceeds to carefully and formally define what a *model* is and what *equivalent* means.

Key observation

Every step of this exercise is motivated by the mathematics, not intuitive reflection on what they are supposed to mean or capture. This is a stark contrast with what Turing did.

Confluence of large carindals, inner models, and determinacy hypotheses

Typical theorem in set theory

The following are equivalent:

1. A particular class of sets of real numbers is determined.
2. There is an inner model for a particular kind of large cardinals.

Confluence of large carindals, inner models, and determinacy hypotheses

Maddy (2011). *Defending the Axioms*

"Considering that determinacy and large cardinals arose in the course of such disparate, apparently unrelated contexts of mathematical inquiry, this ultimate equivalence is quite surprising and impressive: 'This sort of convergence of conceptually distinct domains is striking and unlikely to be an accident' (Koellner, 2006, p. 174). Our second-philosophical Objectivist understands the situation this way: the fact that two apparently fruitful mathematical themes turn out to coincide makes it all the more likely that they're tracking a genuine strain of mathematical depth." (Maddy, 2011, p. 129)

The spirit of Remarkable Coincidence

Maddy (2011). *Defending the Axioms*

Once we have a concept that's mathematically fruitful, it's rational policy to exploit it further, to try to extend it in ways that seem "natural" or harmonious with its leading intuitions ... What's striking is that all these perfectly reasonable ways of proceeding are in fact grounded in their promise of leading to the realization of more of our mathematical goals, to the discovery of more fruitful concepts and theories, to the production of more deep mathematics ... What does matter, all that really matters, is the fruitfulness and promise of the mathematics itself.

Counterexample Resistance: an anecdote

- Kleene, upon hearing Church's Thesis for the first time: "he can't be right" (Crossley, 2006, p. 7).

Counterexample Resistance: an anecdote

- Kleene, upon hearing Church's Thesis for the first time: "he can't be right" (Crossley, 2006, p. 7).
- But he was converted overnight, when he realized he could not diagonalize out of the computable functions.

Counterexample Resistance: an anecdote

- Kleene, upon hearing Church's Thesis for the first time: "he can't be right" (Crossley, 2006, p. 7).
- But he was converted overnight, when he realized he could not diagonalize out of the computable functions.

Counterexample Resistance: an anecdote

- Kleene, upon hearing Church's Thesis for the first time: "he can't be right" (Crossley, 2006, p. 7).
- But he was converted overnight, when he realized he could not diagonalize out of the computable functions.

More of Kleene's argument by confluence

"The exploration of various methods which might be expected to lead to a function outside the class of the general recursive functions has in every case shown either that the method does not actually lead outside."

Counterexample Resistance: an anecdote

- Kleene, upon hearing Church's Thesis for the first time: "he can't be right" (Crossley, 2006, p. 7).
- But he was converted overnight, when he realized he could not diagonalize out of the computable functions.

More of Kleene's argument by confluence

"The exploration of various methods which might be expected to lead to a function outside the class of the general recursive functions has in every case shown either that the method does not actually lead outside."

Gödel

For the concept of computability ... [b]y a kind of miracle it is not necessary to distinguish orders, and the diagonal procedure does not lead outside the defined notion.

Other evidence of Counterexample Resistance

Rogers (1987). *Theory of recursive functions and effective computability*

These equivalence demonstrations can be generalized to show that over certain very broad families of enlargements of these formal characterizations the class of partial functions obtained remains unchanged ... For example, if we allow more than one tape, or other symbols than 1 and B, in the definition of Turing machine, the partial functions obtainable are still partial recursive functions.

Computability is surprisingly resistant against putative enhancements

- (Rósza Péter) Adding course-of-value recursion and nested recursion does not lead to more primitive recursive functions
- (Sacks, Leeuw et al.) Introducing randomness or genericity (in a technical sense) to computability does not lead to more computable functions.

Counterexample Resistance

Confluence ⇒ Putative enhancements that might lead outside of the class of computable functions do not in fact do so. They end up equivalent to the original definitions, despite appearing to be strengthenings. It is as if tipping the scales in favor of the uncomputable does not actually tip the scales at all.

Application: generalized constructibility

Definition (Gödel's constructible hierarchy)

Letting $\mathcal{D}(M)$ be the set of first-order definable subsets of M, we define

$$L_0 = \emptyset$$
$$L_{\alpha+1} = \mathcal{D}(L_\alpha)$$
$$L_\lambda = \bigcup_{\alpha<\lambda} L_\alpha, \text{ for limit } \lambda$$
$$L = \bigcup_{\alpha\in \mathsf{Ord}} L_\alpha$$

Application: generalized constructibility

Definition (Generalized constructible hierarchy)

Letting $\mathcal{D}_{\mathcal{L}}(M)$ be the set of definable subsets of M in the abstract logic $\mathcal{L}$, we define

$$L'_0 = \emptyset$$
$$L'_{\alpha+1} = \mathcal{D}(L'_\alpha)$$
$$L'_\lambda = \bigcup_{\alpha<\lambda} L'_\alpha, \text{ for limit } \lambda$$
$$C(\mathcal{L}) = \bigcup_{\alpha\in \mathsf{Ord}} L'_\alpha$$

Application: generalized constructibility

Theorem

$C(\mathcal{L}) = L$ if $\mathcal{L}$ is first-order logic equipped with any of the following.

1. *cardinality quantifiers Q_α, $\alpha \in \mathrm{Ord}$*
2. *equivalence quantifiers Q^E_α (this quantifier has the meaning "φ defines an equivalence relation with $\geq \aleph_\alpha$-many equivalence classes")*
3. *recursive countable conjunctions and disjunctions*
4. *recursive game quantifiers: $\forall x_0 \exists x_1 \forall x_2 \exists x_3 \ldots \bigwedge_{n\in\omega} \varphi_n(x_0, y_0 \ldots, x_n, y_n)$*
5. *well-ordering quantifier*

Application: generalized constructibility

Kennedy (2013). *On Formalism-freeness*

"We suggest that this manifests a remarkable independence of L from the formalism used ... Constructibility being not particularly sensitive to the underlying logic in that sense gives evidence that a type of Church-Turing thesis holds for L, namely invariance with respect to a certain large class of logics."

Application: generalized constructibility

Kennedy (2013). *On Formalism-freeness*

"We suggest that this manifests a remarkable independence of L from the formalism used ... Constructibility being not particularly sensitive to the underlying logic in that sense gives evidence that a type of Church-Turing thesis holds for L, namely invariance with respect to a certain large class of logics."

Maddy: But what exactly are we to conclude from these observations?

A more precise answer from our framework

Lesson: the generalized constructible hierarchies add expressive resources to first-order logic, the same way Péter, Sacks, and others add computational sources to computability. So, the fact that they end up equivalent to the original constructible hierarchy is a form of Counterexample Resistance.

Claim: Kennedy's result was not joint-carving

Because it lacks any substantial claim to joint-detection. And a more serious worry...

Claim: Kennedy's result was not joint-carving

Because it lacks any substantial claim to joint-detection. And a more serious worry...

Theorem

For "almost every naturally occurring logic" $\mathcal{L}$, there is a class of ordinals A such that $C(\mathcal{L}) = L[A]$.

Claim: Kennedy's result was not joint-carving

Because it lacks any substantial claim to joint-detection. And a more serious worry...

Theorem

For "almost every naturally occurring logic" $\mathcal{L}$, there is a class of ordinals A such that $C(\mathcal{L}) = L[A]$.

Upshot

So looking at these $C(\mathcal{L})$'s is just looking at L "equipped with various oracles"

Claim: Kennedy's result was not joint-carving

Consider the parallel case for computability

After all, we weren't sold on the Church-Turing Thesis by looking at various oracles and seeing that they don't extend the computational powers of Turing machines. Why should we in this case?

A closer parallel...

Theorem (Koepke)

A set of ordinals is ordinal-Turing-computable iff it is ordinal register-computable iff it is in L.

References I

Adleman, L. M., DeMarrais, J., & Huang, M.-D. A. (1997).Quantum Computability. *SIAM Journal on Computing*, *26*(5), 1524–1540. https://doi.org/10.1137/S0097539795293639

Bernstein, E., & Vazirani, U. (1993).Quantum complexity theory. *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, 11–20. https://doi.org/10.1145/167088.167097

Bernstein, E., & Vazirani, U. (1997).Quantum Complexity Theory. *SIAM Journal on Computing*, *26*(5), 1411–1473. https://doi.org/10.1137/S0097539796300921

Crossley, J. N. (2006). Reminiscences of logicians. In *Algebra and logic: Papers from the 1974 summer research institute of the australian mathematical society, monash university, australia* (pp. 1–62). Springer.

References II

Feynman, R. P., Leighton, R. B., & Sands, M. L. (1964). *The Feynman lectures on physics* (Vol. 2: Mainly electromagnetism and matter). Pearson Addison Wesley.

Foreman, M. (2010).Models for measure preserving transformations. *Topology and its Applications*, *157*(8), 1404–1414. https://doi.org/10.1016/j.topol.2009.06.021

Gao, S. (2008, September 3). *Invariant Descriptive Set Theory* (0th ed.). Chapman and Hall/CRC. https://doi.org/10.1201/9781584887942

Hamkins, J. D., & Lewis, A. (2000).Infinite time Turing machines. *Journal of Symbolic Logic*, *65*(2), 567–604. https://doi.org/10.2307/2586556

References III

Kennedy, J. (2020, December 17). *Gödel, Tarski and the Lure of Natural Language: Logical Entanglement, Formalism Freeness* (1st ed.). Cambridge University Press. https://doi.org/10.1017/9780511998393

Kleene, S. C. (1952). *Introduction to Metamathematics*. Van Nostrand.

Koellner, P. (2006).On the Question of Absolute Undecidability†. *Philosophia Mathematica*, *14*(2), 153–188. https://doi.org/10.1093/philmat/nkj009

Leeuw, K. D., Moore, E. F., Shannon, C. E., & Shapiro, N. (1956, December 31). Computability by Probabilistic Machines. In C. E. Shannon & J. McCarthy (Eds.), *Automata Studies. (AM-34)* (pp. 183–212). Princeton University Press. https://doi.org/10.1515/9781400882618-010

References IV

Maddy, P. (2007). *Second philosophy: A naturalistic method*. Oxford University Press.
OCLC: ocm76935638.

Maddy, P. (2011, January 27). *Defending the Axioms: On the Philosophical Foundations of Set Theory*. Oxford University Press.

Montalbán, A. (2019).Martin's conjecture: A classification of the naturally occurring Turing degrees. *Notices Amer. Math. Soc*, *66*(8), 1209–1215.

San Mauro, L. (2018). Church-Turing Thesis, in Practice. In M. Piazza & G. Pulcini (Eds.), *Truth, Existence and Explanation: FilMat 2016 Studies in the Philosophy of Mathematics* (pp. 225–248). Springer International Publishing.
https://doi.org/10.1007/978-3-319-93342-9_13