Shore's computational reverse mathematics

Benedict Eastaugh

University of Bristol

Philosophy and Computation workshop
Lunds universitet
May 13, 2012

A foundational dialectic

Suppose we're committed to a particular foundational programme of limited strength, such as predicativism or finitistic reductionism.

A foundational dialectic

Suppose we're committed to a particular foundational programme of limited strength, such as predicativism or finitistic reductionism.

1. How do we know which theorems we're entitled to assert?

A foundational dialectic

Suppose we're committed to a particular foundational programme of limited strength, such as predicativism or finitistic reductionism.

1. How do we know which theorems we're entitled to assert?
2. How do we know what mathematics we're giving up?

Reverse mathematics can help

If we formalise our foundation in *second order arithmetic*, results in reverse mathematics will let us know which theorems we're entitled to assert and which remain out of reach.

Reverse mathematics can help

If we formalise our foundation in *second order arithmetic*, results in reverse mathematics will let us know which theorems we're entitled to assert and which remain out of reach.

This is done by proving *equivalences* between such theorems and subsystems of second order arithmetic, over a weak base theory.

Syntax and semantics of second order arithmetic

Second order arithmetic is a two-sorted first order system with *number variables* $m, n, i, j, \ldots$ and *set variables* $X, Y, Z, \ldots$ ranging over subsets of the domain.

Syntax and semantics of second order arithmetic

Second order arithmetic is a two-sorted first order system with *number variables* $m, n, i, j, \ldots$ and *set variables* $X, Y, Z, \ldots$ ranging over subsets of the domain.

L_2-structures are models of the first order language of arithmetic extended with a collection of sets for the second order variables to range over:

$$\mathcal{M} = \langle M, S, +, \cdot, <, 0, 1 \rangle$$

where $S \subseteq \mathcal{P}(M)$.

Axioms of second order arithmetic

The axioms of second order arithmetic or Z_2 are the universal closures of the following:

- Basic arithmetic axioms: PA minus induction.

Axioms of second order arithmetic

The axioms of second order arithmetic or Z_2 are the universal closures of the following:

- Basic arithmetic axioms: PA minus induction.
- Induction axiom:

$$(0 \in X \wedge \forall n(n \in X \rightarrow n+1 \in X)) \rightarrow \forall n(n \in X).$$

Axioms of second order arithmetic

The axioms of second order arithmetic or Z_2 are the universal closures of the following:

- Basic arithmetic axioms: PA minus induction.
- Induction axiom:

$$(0 \in X \wedge \forall n(n \in X \rightarrow n+1 \in X)) \rightarrow \forall n(n \in X).$$

- Comprehension scheme:

$$\exists X \forall n(n \in X \leftrightarrow \varphi(n))$$

for all φ with X not free.

Subsystems of second order arithmetic

Subsystems of Z_2 are primarily obtained by restricting the comprehension scheme to particular syntactically defined subclasses.

Subsystems of Z_2	**Defining conditions**
RCA_0	Recursive (Δ^0_1) comprehension
WKL_0	RCA_0 plus weak König's lemma
ACA_0	Arithmetical comprehension
ATR_0	ACA_0 plus arithmetical transfinite recursion
$\Pi^1_1{-}CA_0$	Π^1_1 comprehension

Foundational programmes and the Big Five

The most important subsystems of second order arithmetic, known as the Big Five, formally capture some philosophically-motivated programmes in foundations of mathematics.

Foundational programmes and the Big Five

The most important subsystems of second order arithmetic, known as the Big Five, formally capture some philosophically-motivated programmes in foundations of mathematics.

Foundational programmes	**Subsystems of Z_2**
Constructivism	RCA_0
Finitistic reductionism	WKL_0
Predicativism	ACA_0
Predicative reductionism	ATR_0
Impredicativity	$\Pi^1_1{-}CA_0$

Varieties of induction

The second order induction *axiom* ties the strength of induction to the strength of the comprehension axiom: we can do induction only over those sets we can prove exist.

Varieties of induction

The second order induction *axiom* ties the strength of induction to the strength of the comprehension axiom: we can do induction only over those sets we can prove exist.

Contrast this with the *induction scheme*, each instance of which is a theorem of Z_2:

$$(\varphi(0) \land \forall n(\varphi(n) \rightarrow \varphi(n+1))) \rightarrow \forall n\, \varphi(n)$$

for all formulae φ in the language of second order arithmetic.

Varieties of induction

The second order induction *axiom* ties the strength of induction to the strength of the comprehension axiom: we can do induction only over those sets we can prove exist.

Contrast this with the *induction scheme*, each instance of which is a theorem of Z_2:

$$(\varphi(0) \wedge \forall n(\varphi(n) \rightarrow \varphi(n+1))) \rightarrow \forall n\, \varphi(n)$$

for all formulae φ in the language of second order arithmetic.

Weaker forms of induction can be obtained by restricting this scheme to particular classes such as the Σ^0_1 formulae.

Induction axioms and subsystems of Z_2

Σ^0_1 **induction**	**Induction axiom**	**Full induction scheme**
RCA_0		RCA
WKL_0		WKL
	ACA_0	ACA
	ATR_0	ATR
	$\Pi^1_1-CA_0$	Π^1_1-CA

Computational reverse mathematics

Developed by Richard Shore in two recent papers (Shore 2010, 2011), computational reverse mathematics draws on recursion theory rather than proof theory.

Computational reverse mathematics

Developed by Richard Shore in two recent papers (Shore 2010, 2011), computational reverse mathematics draws on recursion theory rather than proof theory.

It has a two-fold motivation:

- Giving an account of reverse mathematics which most mathematicians will find natural, in computational and construction-oriented terms.

Computational reverse mathematics

Developed by Richard Shore in two recent papers (Shore 2010, 2011), computational reverse mathematics draws on recursion theory rather than proof theory.

It has a two-fold motivation:

- Giving an account of reverse mathematics which most mathematicians will find natural, in computational and construction-oriented terms.
- Extending reverse mathematical analysis from countable structures to uncountable ones.

The main question

Can computational reverse mathematics be used to carry out the foundational analysis outlined at the beginning?

The main question

Can computational reverse mathematics be used to carry out the foundational analysis outlined at the beginning?

To answer this, we first need to look at the details of Shore's programme.

ω-models

Computational reverse mathematics builds on a tradition of looking at ω-*models*, structures which extend the standard model of arithmetic $\mathbb{N}$.

- First order part is the natural numbers $\omega = \{0, 1, 2, \dots\}$.

ω-models

Computational reverse mathematics builds on a tradition of looking at ω-*models*, structures which extend the standard model of arithmetic $\mathbb{N}$.

- First order part is the natural numbers $\omega = \{0, 1, 2, \dots\}$.
- Second order part $\mathcal{C} \subseteq \mathcal{P}(\omega)$ closed under particular recursion-theoretic operations.

ω-models

Computational reverse mathematics builds on a tradition of looking at ω-*models*, structures which extend the standard model of arithmetic $\mathbb{N}$.

- First order part is the natural numbers $\omega = \{0, 1, 2, \dots\}$.
- Second order part $\mathcal{C} \subseteq \mathcal{P}(\omega)$ closed under particular recursion-theoretic operations.
- Closure under more operations $\Leftrightarrow$ model of stronger theories.

Turing ideals

These models are also known as *Turing ideals*.

Turing ideals

These models are also known as *Turing ideals*. All Turing ideals are models of RCA.

Turing ideals

These models are also known as *Turing ideals*. All Turing ideals are models of RCA. Turing ideals satisfying stronger closure conditions are also models of stronger theories such as ACA.

Turing ideals

These models are also known as *Turing ideals*. All Turing ideals are models of RCA. Turing ideals satisfying stronger closure conditions are also models of stronger theories such as ACA.

Definition (Turing ideal)

Let $\mathcal{C}$ be a nonempty subset of $\mathcal{P}(\omega)$ closed under Turing reducibility and recursive joins. Then we call $\mathcal{C}$ a *Turing ideal*.

Turing ideals

These models are also known as *Turing ideals*. All Turing ideals are models of RCA. Turing ideals satisfying stronger closure conditions are also models of stronger theories such as ACA.

Definition (Turing ideal)

Let $\mathcal{C}$ be a nonempty subset of $\mathcal{P}(\omega)$ closed under Turing reducibility and recursive joins. Then we call $\mathcal{C}$ a *Turing ideal*.

A set X is *Turing reducible* to a set Y, $X \leq_T Y$, iff there is a Turing machine with an oracle for Y which computes X.

The *recursive join* of two sets X and Y is given by

$$X \oplus Y = \{2n : n \in X\} \cup \{2n + 1 : n \in Y\}.$$

Closure conditions and subsystems of Z_2

Closure conditions	**Subsystems of Z_2**
Turing reducibility and recursive joins	RCA
Jockush–Soare low basis theorem	WKL
Turing jump	ACA
Hyperarithmetic reducibility	ATR
Hyperjump	Π^1_1–CA

Computable entailment and equivalence

Traditional reverse mathematics looks for *provable equivalences* over a base theory.

Computable entailment and equivalence

Traditional reverse mathematics looks for *provable equivalences* over a base theory. Computational reverse mathematics looks for *computable equivalences*.

Computable entailment and equivalence

Traditional reverse mathematics looks for *provable equivalences* over a base theory. Computational reverse mathematics looks for *computable equivalences*.

Definition (Computable entailment and equivalence)

Let $\mathcal{C}$ be a Turing ideal, and let φ be a sentence of second order arithmetic. $\mathcal{C}$ *computably satisfies* φ if φ is true in the ω-model whose second order part consists of $\mathcal{C}$.

Computable entailment and equivalence

Traditional reverse mathematics looks for *provable equivalences* over a base theory. Computational reverse mathematics looks for *computable equivalences*.

Definition (Computable entailment and equivalence)

Let $\mathcal{C}$ be a Turing ideal, and let φ be a sentence of second order arithmetic. $\mathcal{C}$ *computably satisfies* φ if φ is true in the ω-model whose second order part consists of $\mathcal{C}$.

A sentence ψ *computably entails* φ, $\psi \models_{\mathcal{C}} \varphi$, if every Turing ideal $\mathcal{C}$ satisfying ψ also satisfies φ.

Computable entailment and equivalence

Traditional reverse mathematics looks for *provable equivalences* over a base theory. Computational reverse mathematics looks for *computable equivalences*.

Definition (Computable entailment and equivalence)

Let $\mathcal{C}$ be a Turing ideal, and let φ be a sentence of second order arithmetic. $\mathcal{C}$ *computably satisfies* φ if φ is true in the ω-model whose second order part consists of $\mathcal{C}$.

A sentence ψ *computably entails* φ, $\psi \models_{\mathcal{C}} \varphi$, if every Turing ideal $\mathcal{C}$ satisfying ψ also satisfies φ.

Two sentences ψ and φ are *computably equivalent*, $\psi \equiv_{\mathcal{C}} \varphi$, if each computably entails the other.

Computable entailment and equivalence

Traditional reverse mathematics looks for *provable equivalences* over a base theory. Computational reverse mathematics looks for *computable equivalences*.

Definition (Computable entailment and equivalence)

Let $\mathcal{C}$ be a Turing ideal, and let φ be a sentence of second order arithmetic. $\mathcal{C}$ *computably satisfies* φ if φ is true in the ω-model whose second order part consists of $\mathcal{C}$.

A sentence ψ *computably entails* φ, $\psi \models_{\mathcal{C}} \varphi$, if every Turing ideal $\mathcal{C}$ satisfying ψ also satisfies φ.

Two sentences ψ and φ are *computably equivalent*, $\psi \equiv_{\mathcal{C}} \varphi$, if each computably entails the other.

These definitions extend to theories in the obvious way.

Computable entailment is insensitive to induction

Because computable entailment only considers ω-models, systems with restricted induction are computably equivalent to those with the full second order induction scheme.

Computable entailment is insensitive to induction

Because computable entailment only considers ω-models, systems with restricted induction are computably equivalent to those with the full second order induction scheme.

$$\begin{aligned} \mathsf{RCA}_0 &\equiv_{\mathcal{C}} \mathsf{RCA} \\ \mathsf{WKL}_0 &\equiv_{\mathcal{C}} \mathsf{WKL} \\ \mathsf{ACA}_0 &\equiv_{\mathcal{C}} \mathsf{ACA} \\ &\vdots \end{aligned}$$

Computable entailment is insensitive to induction

Because computable entailment only considers ω-models, systems with restricted induction are computably equivalent to those with the full second order induction scheme.

$$\mathsf{RCA}_0 \equiv_{\mathcal{C}} \mathsf{RCA}$$
$$\mathsf{WKL}_0 \equiv_{\mathcal{C}} \mathsf{WKL}$$
$$\mathsf{ACA}_0 \equiv_{\mathcal{C}} \mathsf{ACA}$$
$$\vdots$$

Why is this problematic? Because the full second order induction scheme is proof-theoretically very strong.

Hilbert's programme

Hilbert's programme was to reduce infinitary mathematics to finitary mathematics.

Hilbert's programme

Hilbert's programme was to reduce infinitary mathematics to finitary mathematics.

This reduction was to be accomplished by giving a finitary consistency proof for an infinitary system which we can identify with Z_2.

Hilbert's programme

Hilbert's programme was to reduce infinitary mathematics to finitary mathematics.

This reduction was to be accomplished by giving a finitary consistency proof for an infinitary system which we can identify with Z_2.

Gödel's second incompleteness theorem shows that there is no such proof. Hilbert's programme therefore cannot be carried out in its entirety.

Partial realisations of Hilbert's programme

Stephen Simpson (1988) has raised the possibility of *partial realisations* of Hilbert's programme.

Partial realisations of Hilbert's programme

Stephen Simpson (1988) has raised the possibility of *partial realisations* of Hilbert's programme.

The main question he addresses is this: how much of infinitary mathematics can we retain in a system which is Π^0_1-conservative over primitive recursive arithmetic (PRA)?

Partial realisations of Hilbert's programme

Stephen Simpson (1988) has raised the possibility of *partial realisations* of Hilbert's programme.

The main question he addresses is this: how much of infinitary mathematics can we retain in a system which is Π^0_1-conservative over primitive recursive arithmetic (PRA)?

Theorem (Friedman)

WKL_0 *is* Π^0_2*-conservative over* PRA.

Partial realisations of Hilbert's programme

Stephen Simpson (1988) has raised the possibility of *partial realisations* of Hilbert's programme.

The main question he addresses is this: how much of infinitary mathematics can we retain in a system which is Π^0_1-conservative over primitive recursive arithmetic (PRA)?

Theorem (Friedman)

WKL_0 *is* Π^0_2*-conservative over* PRA.

Theorem (Sieg)

PRA *proves that* WKL_0 *is* Π^0_2*-conservative over* PRA.

WKL is not finitistically reducible

Theorem

WKL $\vdash \mathrm{Con}(\mathrm{PRA})$.

WKL is not finitistically reducible

Theorem

WKL $\vdash \mathrm{Con}(\mathrm{PRA})$.

$\mathrm{Con}(\mathrm{PRA})$ is a Π^0_1 statement not provable in PRA

WKL is not finitistically reducible

Theorem

WKL $\vdash \mathrm{Con}(\mathrm{PRA})$.

$\mathrm{Con}(\mathrm{PRA})$ is a Π^0_1 statement not provable in PRA, so WKL is not Π^0_1-conservative over PRA

WKL is not finitistically reducible

Theorem

WKL $\vdash \mathrm{Con}(\mathrm{PRA})$.

$\mathrm{Con}(\mathrm{PRA})$ is a Π^0_1 statement not provable in PRA, so WKL is not Π^0_1-conservative over PRA and therefore not finitistically reducible to it.

Computable entailment does not preserve justification

A key property of any entailment relation is preserving justification: if we are justified in accepting the antecedent then we are also justified in accepting the consequent.

Computable entailment does not preserve justification

A key property of any entailment relation is preserving justification: if we are justified in accepting the antecedent then we are also justified in accepting the consequent.

By their own lights a finitistic reductionist will be justified in accepting the Π^0_1 sentences φ proved by WKL_0

Computable entailment does not preserve justification

A key property of any entailment relation is preserving justification: if we are justified in accepting the antecedent then we are also justified in accepting the consequent.

By their own lights a finitistic reductionist will be justified in accepting the Π^0_1 sentences φ proved by WKL_0 but *not* those proved by WKL, since WKL is not finitistically reducible.

Computable entailment does not preserve justification

A key property of any entailment relation is preserving justification: if we are justified in accepting the antecedent then we are also justified in accepting the consequent.

By their own lights a finitistic reductionist will be justified in accepting the Π^0_1 sentences φ proved by WKL_0 but *not* those proved by WKL, since WKL is not finitistically reducible.

But $\mathsf{WKL}_0 \models_{\mathcal{C}}$ WKL, so computable entailment does not preserve justification within the foundational programmes it seeks to analyse.

Conclusion

Computational reverse mathematics doesn't respect the justificatory structure of foundational programmes.

Conclusion

Computational reverse mathematics doesn't respect the justificatory structure of foundational programmes.

So whatever its merits, Shore's framework doesn't seem suitable for the kind of foundational analysis outlined at the beginning of the talk.

Thank you.

References

R. A. Shore. Reverse Mathematics: The Playground of Logic. *The Bulletin of Symbolic Logic*, Volume 16, Number 3, 2010.

R. A. Shore. Reverse mathematics, countable and uncountable: a computational approach. Manuscript, 2011.

S. G. Simpson. Partial realizations of Hilbert's program. *The Journal of Symbolic Logic*, 53:349–363, 1988.