Finitism distilled

Daniel Leivant

Indiana University

Finitism

Finite vs infinite

- Infinite is good:
  - Ingrained in spacial intuition
  - Required in formalization (calculus)
  - Essential to generality of mathematics (Poincaré)
  - Central in abstractions (functionals...)

Finite vs infinite

- Infinite is good:
  - Ingrained in spacial intuition
  - Required in formalization (calculus)
  - Essential to generality of mathematics (Poincaré)
  - Central in abstractions (functionals...)
- BUT not without reservation:
  - Infinity is at the core of paradoxes
  - The universe is likely finite
  - Spacial continuity is a deception

Finite vs infinite

- Infinite is good:
    - Ingrained in spacial intuition
    - Required in formalization (calculus)
    - Essential to generality of mathematics (Poincaré)
    - Central in abstractions (functionals...)
- BUT not without reservation:
    - Infinity is at the core of paradoxes
    - The universe is likely finite
    - Spacial continuity is a deception
- Infinity is essential, finiteness remains fundamental.

Hilbert's Program

- Response to the Foundations Crisis:
- Trying to have it both ways.
  ”Infinity is just a figure of speech”:
  Show by finitistic means that mathematics,
  actual infinite and all, is at least consistent

Hilbert's Program

- Response to the Foundations Crisis:
- Trying to have it both ways.
  ”Infinity is just a figure of speech”:
  Show by finitistic means that mathematics,
  actual infinite and all, is at least consistent
- Smashed by Gödel: Consistency unprovable, let alone using just finitistic means

Hilbert's Program

- Response to the Foundations Crisis:
- Trying to have it both ways.
  "Infinity is just a figure of speech":
  Show by finitistic means that mathematics,
    actual infinite and all, is at least consistent
- Smashed by Gödel: Consistency unprovable, let alone using just finitistic means
- A MIRROR PROJECT (Tait):
    Delineate the finitistic core of mathematics
    without recourse to the infinite.

Delineating finitism

- Hilbert/Bernays: Admit at least Primitive Recursive Mathematics.

Delineating finitism

- Hilbert/Bernays: Admit at least Primitive Recursive Mathematics.

  What's that?

Delineating finitism

- Hilbert/Bernays: Admit at least Primitive Recursive Mathematics.
- $\boxed{\textit{Recurrence}}$ over $\mathbb{N}$ (Skolem 1932):
  Define $f : \mathbb{N} \times X \to X$ from g_0 and g_s.

$$\boxed{\begin{array}{rcl} f(0, X) & = & g_0(X) \\ f(\mathrm{s}(n), X) & = & g_s(f(n, X), X, n) \end{array}}$$

Delineating finitism

- Hilbert/Bernays: Admit at least Primitive Recursive Mathematics.
- **Recurrence** over $\mathbb{N}$ (Skolem 1932):
  Define $f : \mathbb{N} \times X \to X$ from g_0 and g_s.

$$\boxed{\begin{aligned} f(0, X) &= g_0(X) \\ f(\mathrm{s}(n), X) &= g_s(f(n, X), X, n) \end{aligned}}$$

- + defined using $X = \mathbb{N}$ from 0 and successor.
  Using $X = N \times (N \to N)$ yields Ackermann's function

Delineating finitism

- Hilbert/Bernays: Admit at least Primitive Recursive Mathematics.
- **Recurrence** over $\mathbb{N}$ (Skolem 1932):
  Define $f : \mathbb{N} \times X \to X$ from g_0 and g_s.

$$\begin{array}{rcl} f(0, X) & = & g_0(X) \\ f(\mathrm{s}(n), X) & = & g_s(f(n, X), X, n) \end{array}$$

- + defined using $X = \mathbb{N}$ from 0 and successor.
  Using $X = N \times (N \to N)$ yields Ackermann's function
- **PR functions:** Generated from constructors using recurrence for $X = \mathbb{N}^k$ and explicit definitions.

Primitive Recursive Arithmetic

- A formal mathematical theory for the PR world.
- Symbols and defining equations for all PR functions.
- Induction template:
  $\boxed{(\forall n\, \varphi[n] \to \varphi[sn]) \;\to\; \varphi[0] \to \varphi[x]}$ φ quantifier-free

Primitive Recursive Arithmetic

- A formal mathematical theory for the PR world.
- Symbols and defining equations for all PR functions.
- Induction template:
  $\boxed{(\forall n\, \varphi[n] \to \varphi[\mathrm{s}n]) \;\to\; \varphi[0] \to \varphi[x]}$ φ quantifier-free
- Hilbert/Bernays admitted PR, didn't claim it exhausts finitism.
- Some in the 1990s:
  Induction is finitistic. Why can't φ be any formula?

Primitive Recursive Arithmetic

- A formal mathematical theory for the PR world.
- Symbols and defining equations for all PR functions.
- Induction template:
  $\boxed{(\forall n\, \varphi[n] \rightarrow \varphi[sn]) \;\rightarrow\; \varphi[0] \rightarrow \varphi[x]}$ φ quantifier-free
- Hilbert/Bernays admitted PR, didn't claim it exhausts finitism.
- Some in the 1990s:
  Induction is finitistic. Why can't φ be any formula?
- Tait (2002,2012): φ should be predicative:
  whereas $\forall$ in φ presupposes $\mathbb{N}$ as totality.
- ***Tait's Thesis:*** **Finitistic Mathematics is precisely PRA.**

Articulating finitism explicitly

- Looking closer, even formal PRA is not finitistic:
  it refers to PR functions, i.e. infinite objects.
- Something more is needed to justify Tait's Thesis.

Articulating finitism explicitly

- Looking closer, even formal PRA is not finitistic:
  it refers to PR functions, i.e. infinite objects.
- Something more is needed to justify Tait's Thesis.
- If finitism should stand on its own feet,
  it should be built up, not down.
- Focusing on finite sets impedes basic data-changes.
  We take ***finite functions (ff's)*** of arity 0,1,2 over ***"atoms."***

Finite data systems (FDS)

- ***Vocabulary:*** A finite set V of fnc-ids of arity 0,1, or 2.

Finite data systems (FDS)

- $\boxed{\textit{Vocabulary:}}$ A finite set V of fnc-ids of arity 0,1, or 2.
- $\boxed{V\textit{-FDS}}$ (Finite Data System):
  A mapping assigning to each $\mathbf{f} \in V$
  a finite-function (ff) of corresponding arity.

Finite data systems (FDS)

- $\boxed{\textit{Vocabulary:}}$ A finite set V of fnc-ids of arity 0,1, or 2.
- $\boxed{V\textit{-FDS}}$ (Finite Data System):
  A mapping assigning to each $\mathbf{f} \in V$
  a finite-function (ff) of corresponding arity.

  A $\boxed{V\textit{-molecule}}$ is a W-FDS for some $W \subseteq V$.

Finite data systems (FDS)

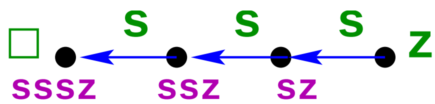
- $\boxed{\textit{Vocabulary:}}$ A finite set V of fnc-ids of arity 0,1, or 2.
- $\boxed{V\textit{-FDS}}$ (Finite Data System):
  A mapping assigning to each $\mathbf{f} \in V$
  a finite-function (ff) of corresponding arity.

  A $\boxed{V\textit{-molecule}}$ is a W-FDS for some $W \subseteq V$.
- The $\boxed{\textit{universe}}$ of a FDS is
  the union of the domains and co-domains of its ff's.

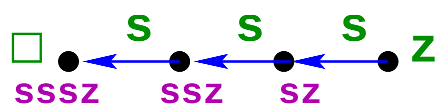
Denotations

- $\boxed{V\text{-terms}}$ are generated from V as usual.
  A reserved id ω denotes "undefined".
  Terms without it are $\boxed{\textit{standard.}}$
- A standard V-term t has a $\boxed{\textit{value}}$ t_σ in a V-molecule σ.

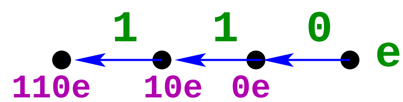
Examples: Molecules for numbers and strings

Natural number 3: 

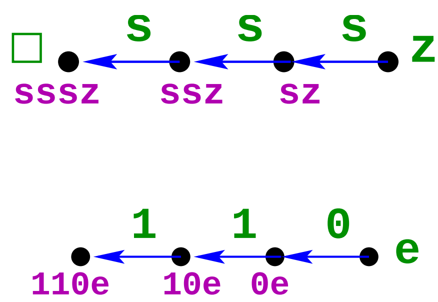
Examples: Molecules for numbers and strings



String 110

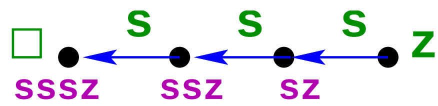


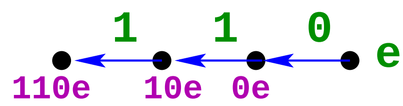
Examples: Molecules for numbers and strings



- The box is an optional pointer to refer to the molecule.

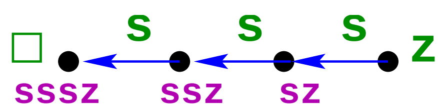
Examples: Molecules for numbers and strings

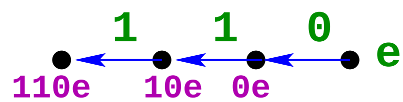




- The box is an optional pointer to refer to the molecule.
- Each example is an FDS, as is their union.
  Considering the union as two molecules is optional.

Examples: Molecules for numbers and strings





- The box is an optional pointer to refer to the molecule.
- Each example is an FDS, as is their union.
  Considering the union as two molecules is optional.
- A second box would require a fresh id.

Computing with FDSs

Updates

- A generic basic operation.
- *Updates:* $\boxed{\mathbf{f}\,\mathbf{t}_1 \cdots \mathbf{t}_k := \mathbf{q}}$

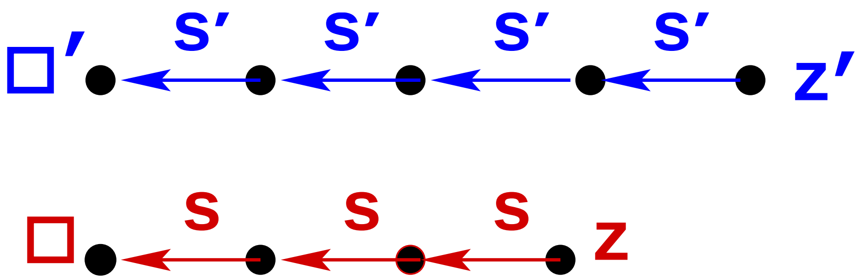
Updates

- A generic basic operation.
- *Updates:* $\boxed{\mathbf{f}\,\mathbf{t}_1 \cdots \mathbf{t}_k := \mathbf{q}}$
- Special cases:
  - *Contractions:* $\boxed{\mathbf{f}\,\mathbf{t}_1 \cdots \mathbf{t}_k := \omega}$

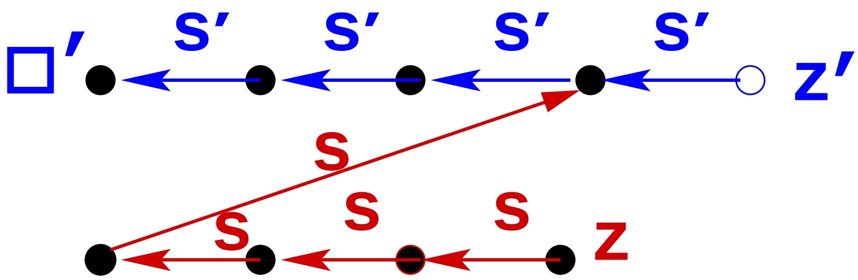
Updates

- A generic basic operation.
- *Updates:* $\boxed{\mathbf{f}\,\mathbf{t}_1 \cdots \mathbf{t}_k := \mathbf{q}}$
- Special cases:
  - *Contractions:* $\boxed{\mathbf{f}\,\mathbf{t}_1 \cdots \mathbf{t}_k := \omega}$
  - *Inceptions:* $\boxed{\mathbf{c} := \mathbf{q} \ \text{ where } \ \sigma(\mathbf{c}) = \bot}$

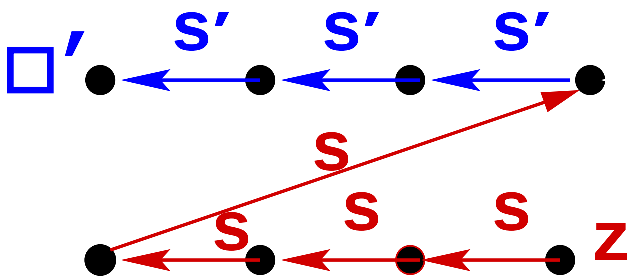
Example: A concatenation algorithm



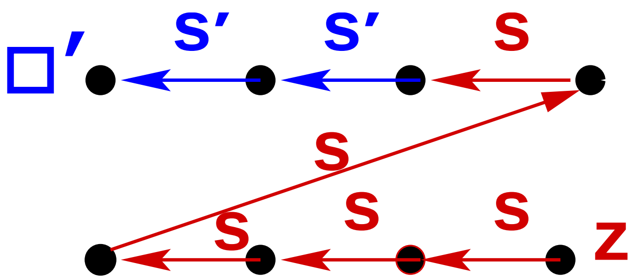
Example: A concatenation algorithm



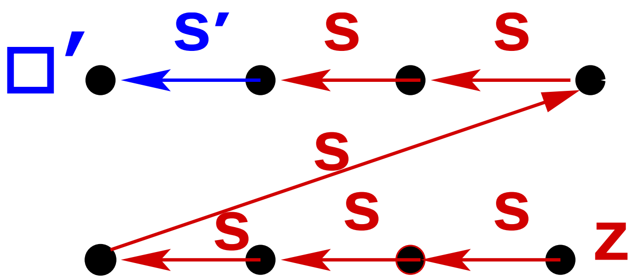
Example: A concatenation algorithm



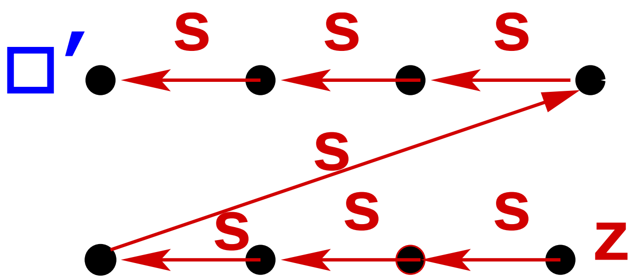
Example: A concatenation algorithm



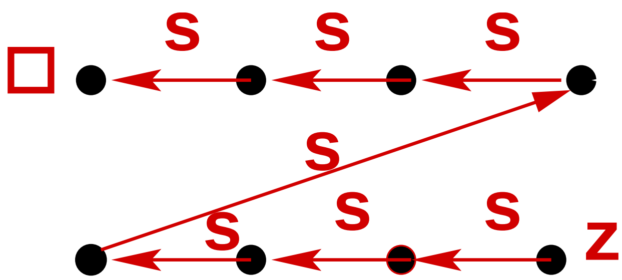
Example: A concatenation algorithm



Example: A concatenation algorithm



Example: A concatenation algorithm



Imperative programs

- Here is an imperative programming language
  that best serves our principle proof-theoretic goals.

Imperative programs

- Here is an imperative programming language
  that best serves our principle proof-theoretic goals.
- Programs are generated from ***updates*** using
  - Composition: $P; Q$

Imperative programs

- Here is an imperative programming language
  that best serves our principle proof-theoretic goals.
- Programs are generated from ***updates*** using
  - Composition: $P; Q$
  - Branching: $\mathbf{if}\,[\mathbf{t} = \mathbf{q}]\,\{P\}\,\{Q\}$

Imperative programs

- Here is an imperative programming language that best serves our principle proof-theoretic goals.
- Programs are generated from ***updates*** using
  - Composition: $P; Q$
  - Branching: $\mathbf{if}\,[\mathbf{t} = \mathbf{q}]\,\{P\}\,\{Q\}$
  - Iteration: $\mathbf{do}\ [T]\{P\} \quad (T \subset V)$
- Semantics: Iteration repeats $\leqslant |\sigma(T)|$ times.

Programs

- **Theorem.**
  A fnct over $\mathbb{N}$ is computable by a program iff it is PR.

Programs

- **Theorem.**
  A fnct over $\mathbb{N}$ is computable by a program iff it is PR.
- $\Rightarrow$: Every program over molecules for $\mathbb{N}$
  yields a PR function because the semantics of programs is coded in PRA.

Programs

- **Theorem.**
  A fnct over $\mathbb{N}$ is computable by a program iff it is PR.
- $\Rightarrow$: Every program over molecules for $\mathbb{N}$ yields a PR function because the semantics of programs is coded in PRA.
- $\Leftarrow$: Every PR is computable by a program because the recurrence schema is implementable by a loop with a counter.

Concrete proof theory

A formal language

- A-variables $u, v, \ldots$ for atoms.
  ff-variables $f, g, \ldots$ for ffs.

A formal language

- A-variables $u, v, \ldots$ for atoms.
  ff-variables $f, g, \ldots$ for ffs.
- $\boxed{\textit{Formulas:}}$ Generated from equations between terms using connectives & quantifiers

A formal language

- A-variables $u, v, \ldots$ for atoms.
  ff-variables $f, g, \ldots$ for ffs.
- **Formulas:** Generated from equations between terms using connectives & quantifiers
  - **Elementary** formulas: No ff-quantifiers

A formal language

- A-variables $u, v, \ldots$ for atoms.
  ff-variables $f, g, \ldots$ for ffs.
- $\boxed{\textit{Formulas:}}$ Generated from equations between terms using connectives & quantifiers
  - $\boxed{\textit{Elementary}}$ formulas: No ff-quantifiers
  - $\boxed{\textit{Concrete}}$ formulas:
    Also allowing ***positively-occurring*** $\exists^f$.

  E.g. $\exists f_1 \ldots f_k\, \psi$, ψ elementary.

A formal language

- A-variables $u, v, \ldots$ for atoms.
  ff-variables $f, g, \ldots$ for ffs.
- **Formulas:** Generated from equations between terms using connectives & quantifiers
  - **Elementary** formulas: No ff-quantifiers
  - **Concrete** formulas:
    Also allowing ***positively-occurring*** $\exists^f$.

  E.g. $\exists f_1 \ldots f_k\, \psi$, ψ elementary.
- Concrete formulas have no reference to infinities.
  Occurrence of $\exists$-ff points to an un-named function!

Some concrete formulas

- Molecule $\boxed{\textit{isomorphism}}$ is definable by a concrete formula.

Some concrete formulas

- Molecule ***isomorphism*** is definable by a concrete formula.
- The molecules representing ***natural numbers*** are definable by the conjunction of
  - The separation properties
    $(\forall u, v\ (su = sv \;\to\; u = v)$ and $su \neq \mathrm{z}$
  - Totality $(u = \mathrm{z} \;\vee\; \exists v\ sv = u)$
  - Linearity:
    Existence of a linear order, extending s, on the domain of z, s.

Some concrete formulas

- Molecule $\boxed{\textit{isomorphism}}$ is definable by a concrete formula.
- The molecules representing $\boxed{\textit{natural numbers}}$ are definable by the conjunction of
  - The separation properties
    $(\forall u, v\ (su = sv \ \rightarrow\ u = v)$ and $su \neq z$
  - Totality $(u = z \ \vee\ \exists v\ sv = u)$
  - Linearity:
    Existence of a linear order, extending s, on the domain of z, s.
- $\boxed{\textit{Inequality}}$ on $\mathbb{N}$ is definable as the existence of a non-surjective embedding.

A concrete theory of FDSs

Here is a theory for FDSs
referring only to concrete formulas.

- ***Empty-ff*** $\boxed{\exists f\,\forall u_1 \ldots u_k\; f u_1 \cdots u_k \;=\; \omega}$

A concrete theory of FDSs

Here is a theory for FDSs
referring only to concrete formulas.

- ***Empty-ff*** $\boxed{\exists f\ \forall u_1 \ldots u_k\ f u_1 \cdots u_k \ = \ \omega}$
- ***Strictness*** $\boxed{u_i = \omega \ \ \rightarrow \ \ f u_1 \cdots u_k = \omega}$

A concrete theory of FDSs

Here is a theory for FDSs
referring only to concrete formulas.

- ***Empty-ff*** $\boxed{\exists f\ \forall u_1 \ldots u_k\ f u_1 \cdots u_k \ = \ \omega}$
- ***Strictness*** $\boxed{u_i = \omega \ \ \rightarrow \ \ f u_1 \cdots u_k = \omega}$
- ***Update*** $\boxed{\exists g\ g(\vec{u}) = v \ \ \wedge\ \forall \vec{x}\ \vec{x} \neq \vec{u} \ \rightarrow \ g(\vec{x}) = f(\vec{x})}$
  Definitional extension: $f_{\vec{u}!v}$.

A concrete theory of FDSs

Here is a theory for FDSs
referring only to concrete formulas.

- ***Empty-ff*** $\boxed{\exists f\ \forall u_1 \ldots u_k\ f u_1 \cdots u_k \ = \ \boldsymbol{\omega}}$
- ***Strictness*** $\boxed{u_i = \omega \ \ \rightarrow \ \ f u_1 \cdots u_k = \omega}$
- ***Update*** $\boxed{\exists g\ g(\vec{u}) = v \ \ \wedge \ \forall \vec{x}\ \vec{x} \neq \vec{u} \ \rightarrow \ g(\vec{x}) = f(\vec{x})}$
  Definitional extension: $f_{\vec{u}!v}$.
- ***Unboundedness*** $\boxed{\exists \vec{u}\ \ f^k \vec{u} = \boldsymbol{\omega}}$

The induction rule

- ***Concrete-Induction-Rule*** For φ concrete

$$\frac{\vdash \varphi[\emptyset] \qquad\qquad \vdash \varphi[f] \;\rightarrow\; \varphi[f_{\vec{u}!v}]}{\varphi[j]}$$

The induction rule

- ***Concrete-Induction-Rule*** For φ concrete

$$\frac{\vdash \varphi[\emptyset] \qquad \vdash \varphi[f] \;\rightarrow\; \varphi[f_{\vec{u}!v}]}{\varphi[j]}$$

- When φ is $\exists \vec{g}\, \varphi_0[f, g]$ (φ_0 elementary)
  the second premise is concrete, since it is equivalent to

$$\varphi_0[f, h] \rightarrow \exists g \;\; \mathbf{dfn}(g) \;\wedge\; \varphi_0[f, g] \qquad (h \text{ fresh})$$

The induction rule

- ***Concrete-Induction-Rule*** For φ concrete

$$\frac{\vdash \varphi[\emptyset] \qquad \vdash \varphi[f] \;\to\; \varphi[f_{\vec{u}!v}]}{\varphi[j]}$$

- When φ is $\exists \vec{g}\, \varphi_0[f,g]$ (φ_0 elementary)
  the second premise is concrete, since it is equivalent to

$$\varphi_0[f,h] \to \exists g \;\; \mathbf{dfn}(g) \;\wedge\; \varphi_0[f,g] \qquad (h \text{ fresh})$$

- The induction ***axiom*** for concrete φ is ***not*** concrete:
  The premise is of the Induction Schema is

$$\forall f\; \forall \vec{u}, v\; \varphi[f] \to \varphi[f_{\vec{u}!v}]$$

so $\forall$ counts as a positive existential in the schema
but its scope is not concrete.

The crux

FDS is complete for PRA

- Recall: PRA is quantifier-free induction over PR-functions.

- We show that PRA is interpretable in FDS, i.e.:
  1. PR functions are represented by concrete FDS-formulas
  2. Quantifier-free formulas of PRA are interpreted by concrete FDS-formulas.
  3. The interpretation of PRA induction for these formulas is derived from FDS induction.

Representing PR functions

- Every PR function is computable by an FDS-program, and therefore representable by a concrete FDS-formula.
  - Proved by discourse induction on the PR definition.
  - Each step uses formal FDS-induction to prove the existence of a computation trace.

Interpreting quantifier-free numeric formulas

- PRA equality is represented in FDS by ***isomorphism***.
- Isomorphism and numeric-inequality
  are definable by concrete formulas,
  so qf formulas of PRA are interpreted in FDS
  by concrete formulas.

Interpreting PRA induction

- By Parson's Theorem PRA induction is captured by the Σ_1 induction ***rule***.
- Since $\mathbb{N}$ is definable by a concrete formula,
  Σ_1 formulas are interpreted by concrete formulas.

Conversely, FDS is sound for PRA

- Since all ff's are codable in PRA as numbers,
  FDS is interpretable in PRA,
  with concrete fmls interpreted as existential fmls of PRA,
  and so admissible in PRA by Parson's Theorem.

In a word...

We focused on the “hardware” of finitism,

constructed a first-order theory of that hardware,

and showed that it is proof-theoretically equivalent

to primitive recursive arithmetic.

In a word, Tait’s Thesis is vindicated!