Bounded arithmetic for simpler proof assistants

Paweł Balawender

University of Warsaw

September 4th, 2025

Motivation

- Proof assistants use strong foundations (HOL, CIC, choice, quotients).

Motivation

- Proof assistants use strong foundations (HOL, CIC, choice, quotients).
- Most theorems need far less.

Motivation

- Proof assistants use strong foundations (HOL, CIC, choice, quotients).
- Most theorems need far less.
- Costs of strength:

Motivation

- Proof assistants use strong foundations (HOL, CIC, choice, quotients).
- Most theorems need far less.
- Costs of strength:
  - Huge tactic space -> harder search.

Motivation

- Proof assistants use strong foundations (HOL, CIC, choice, quotients).
- Most theorems need far less.
- Costs of strength:
  - Huge tactic space -> harder search.
  - Loss of computational content (e.g., `noncomputable`).

Motivation

- Proof assistants use strong foundations (HOL, CIC, choice, quotients).
- Most theorems need far less.
- Costs of strength:
  - Huge tactic space -> harder search.
  - Loss of computational content (e.g., `noncomputable`).
- Reverse mathematics seeks to determine which axioms are actually needed

Motivation

- Proof assistants use strong foundations (HOL, CIC, choice, quotients).
- Most theorems need far less.
- Costs of strength:
  - Huge tactic space -> harder search.
  - Loss of computational content (e.g., `noncomputable`).
- Reverse mathematics seeks to determine which axioms are actually needed
- **Aim:** formalize theorems in the weakest adequate system.

Bounded arithmetic

Bounded arithmetic studies some of the weakest arithmetical theories. Here, we will consider $I\Delta_0$, V^0.

Bounded arithmetic

Bounded arithmetic studies some of the weakest arithmetical theories. Here, we will consider $I\Delta_0, V^0$.

At the bottom:

Bounded arithmetic

Bounded arithmetic studies some of the weakest arithmetical theories. Here, we will consider $I\Delta_0$, V^0.

At the bottom:

- you are not able to prove the Pigeonhole Principle ($V^0 \nvdash PHP$)

Bounded arithmetic

Bounded arithmetic studies some of the weakest arithmetical theories. Here, we will consider $I\Delta_0$, V^0.

At the bottom:

- you are not able to prove the Pigeonhole Principle ($V^0 \nvdash PHP$)
- nor that the exponential function is total! ($I\Delta_0 \nvdash \forall x \exists! y \ \exp(x, y)$)

Bounded arithmetic

Bounded arithmetic studies some of the weakest arithmetical theories. Here, we will consider $I\Delta_0, V^0$.

At the bottom:

- you are not able to prove the Pigeonhole Principle ($V^0 \nvdash PHP$)
- nor that the exponential function is total! ($I\Delta_0 \nvdash \forall x \exists! y \ \exp(x, y)$)

Bounded arithmetic

Bounded arithmetic studies some of the weakest arithmetical theories. Here, we will consider $I\Delta_0, V^0$.

At the bottom:

- you are not able to prove the Pigeonhole Principle ($V^0 \nvdash PHP$)
- nor that the exponential function is total! ($I\Delta_0 \nvdash \forall x \exists! y \ \exp(x, y)$)

You need to explicitly add strength, then can:

Bounded arithmetic

Bounded arithmetic studies some of the weakest arithmetical theories. Here, we will consider $I\Delta_0$, V^0.

At the bottom:

- you are not able to prove the Pigeonhole Principle ($V^0 \nvdash PHP$)
- nor that the exponential function is total! ($I\Delta_0 \nvdash \forall x \exists! y \ \exp(x, y)$)

You need to explicitly add strength, then can:

- prove properties of binary addition ($I\Delta_0 \vdash \forall x \forall y \ x + y = y + x$)

Bounded arithmetic

Bounded arithmetic studies some of the weakest arithmetical theories. Here, we will consider $I\Delta_0, V^0$.

At the bottom:

- you are not able to prove the Pigeonhole Principle ($V^0 \nvdash PHP$)
- nor that the exponential function is total! ($I\Delta_0 \nvdash \forall x \exists ! y \ \exp(x, y)$)

You need to explicitly add strength, then can:

- prove properties of binary addition ($I\Delta_0 \vdash \forall x \forall y \ x + y = y + x$)
- define a sorting function

Bounded arithmetic

Bounded arithmetic studies some of the weakest arithmetical theories. Here, we will consider $I\Delta_0$, V^0.

At the bottom:

- you are not able to prove the Pigeonhole Principle ($V^0 \nvdash PHP$)
- nor that the exponential function is total! ($I\Delta_0 \nvdash \forall x \exists! y \ \exp(x, y)$)

You need to explicitly add strength, then can:

- prove properties of binary addition ($I\Delta_0 \vdash \forall x \forall y \ x + y = y + x$)
- define a sorting function
- prove standard graph theorems.

The goals of this presentation

1. Why formalize arithmetic?

These theories correspond nicely to complexity classes.

We want to formalize theorems of the form $I\Delta_0 \vdash \phi(x, y)$ to explore computational contents of the proofs.

2. Demonstrate that it is possible to formalize it

The syntax of our theory: what it "$\phi(x, y)$"? Vocabulary

First, this is our vocabulary (think of them just as some UTF8 symbols, no meaning at all):

- variable names $(x, y, z, \ldots)$

The syntax of our theory: what it "$\phi(x, y)$"? Vocabulary

First, this is our vocabulary (think of them just as some UTF8 symbols, no meaning at all):

- variable names $(x, y, z, \ldots)$
- logical connectives $(\neg, \wedge, \vee)$ and constants $(\bot, \top)$

The syntax of our theory: what it "$\phi(x, y)$"? Vocabulary

First, this is our vocabulary (think of them just as some UTF8 symbols, no meaning at all):

- variable names ($x, y, z, \ldots$)
- logical connectives ($\neg, \wedge, \vee$) and constants ($\bot, \top$)
- quantifiers ($\forall, \exists$)

The syntax of our theory: what it "$\phi(x, y)$"? Vocabulary

First, this is our vocabulary (think of them just as some UTF8 symbols, no meaning at all):

- variable names ($x, y, z, \ldots$)
- logical connectives ($\neg, \wedge, \vee$) and constants ($\bot, \top$)
- quantifiers ($\forall, \exists$)
- parentheses

The syntax of our theory: what it "$\phi(x, y)$"? Vocabulary

First, this is our vocabulary (think of them just as some UTF8 symbols, no meaning at all):

- variable names ($x, y, z, \dots$)
- logical connectives ($\neg, \wedge, \vee$) and constants ($\bot, \top$)
- quantifiers ($\forall, \exists$)
- parentheses
- **function symbols**:

The syntax of our theory: what it "$\phi(x, y)$"? Vocabulary

First, this is our vocabulary (think of them just as some UTF8 symbols, no meaning at all):

- variable names $(x, y, z, \ldots)$
- logical connectives $(\neg, \wedge, \vee)$ and constants $(\bot, \top)$
- quantifiers $(\forall, \exists)$
- parentheses
- **function symbols**:
    - zero-ary: 0, 1,

The syntax of our theory: what it "$\phi(x, y)$"? Vocabulary

First, this is our vocabulary (think of them just as some UTF8 symbols, no meaning at all):

- variable names ($x, y, z, \dots$)
- logical connectives ($\neg, \wedge, \vee$) and constants ($\bot, \top$)
- quantifiers ($\forall, \exists$)
- parentheses
- **function symbols**:
  - zero-ary: 0, 1,
  - binary: addition ($+$), multiplication ($\cdot$)

The syntax of our theory: what it “$\phi(x, y)$”? Vocabulary

First, this is our vocabulary (think of them just as some UTF8 symbols, no meaning at all):

- variable names ($x, y, z, \ldots$)
- logical connectives ($\neg, \wedge, \vee$) and constants ($\bot, \top$)
- quantifiers ($\forall, \exists$)
- parentheses
- **function symbols**:
  - zero-ary: 0, 1,
  - binary: addition ($+$), multiplication ($\cdot$)
- **relation symbols**:

The syntax of our theory: what it "$\phi(x, y)$"? Vocabulary

First, this is our vocabulary (think of them just as some UTF8 symbols, no meaning at all):

- variable names $(x, y, z, \dots)$
- logical connectives $(\neg, \wedge, \vee)$ and constants $(\bot, \top)$
- quantifiers $(\forall, \exists)$
- parentheses
- **function symbols**:
  - zero-ary: 0, 1,
  - binary: addition $(+)$, multiplication $(\cdot)$
- **relation symbols**:
  - binary: $=$, $\leq$

The syntax of our theory: what it “$\phi(x, y)$”? Vocabulary

First, this is our vocabulary (think of them just as some UTF8 symbols, no meaning at all):

- variable names $(x, y, z, \ldots)$
- logical connectives $(\neg, \wedge, \vee)$ and constants $(\perp, \top)$
- quantifiers $(\forall, \exists)$
- parentheses
- **function symbols**:
  - zero-ary: 0, 1,
  - binary: addition $(+)$, multiplication $(\cdot)$
- **relation symbols**:
  - binary: $=$, $\leq$

The syntax of our theory: what it "$\phi(x, y)$"? Vocabulary

First, this is our vocabulary (think of them just as some UTF8 symbols, no meaning at all):

- variable names $(x, y, z, \ldots)$
- logical connectives $(\neg, \wedge, \vee)$ and constants $(\bot, \top)$
- quantifiers $(\forall, \exists)$
- parentheses
- **function symbols**:
  - zero-ary: 0, 1,
  - binary: addition $(+)$, multiplication $(\cdot)$
- **relation symbols**:
  - binary: $=$, $\leq$

Technicality: require the $=$ symbol be the *actual* equality on underlying objects. Will skip equality axioms later.

The syntax of our theory: what it "$\phi(x, y)$"? Terms and formulas

Terms:

- every variable is a term

The syntax of our theory: what it "$\phi(x, y)$"? Terms and formulas

Terms:

- every variable is a term
- $0, 1$ are terms

The syntax of our theory: what it "$\phi(x, y)$"? Terms and formulas

Terms:

- every variable is a term
- $0, 1$ are terms
- if t_1, t_2 are terms, then $t_1 + t_2$ and $t_1 \cdot t_2$ are terms.

The syntax of our theory: what it "$\phi(x, y)$"? Terms and formulas

Terms:

- every variable is a term
- $0, 1$ are terms
- if t_1, t_2 are terms, then $t_1 + t_2$ and $t_1 \cdot t_2$ are terms.

The syntax of our theory: what it "$\phi(x, y)$"? Terms and formulas

Terms:

- every variable is a term
- $0, 1$ are terms
- if t_1, t_2 are terms, then $t_1 + t_2$ and $t_1 \cdot t_2$ are terms.

Formulas:

- $\perp, \top$ are formulas

The syntax of our theory: what it "$\phi(x, y)$"? Terms and formulas

Terms:

- every variable is a term
- $0, 1$ are terms
- if t_1, t_2 are terms, then $t_1 + t_2$ and $t_1 \cdot t_2$ are terms.

Formulas:

- $\perp, \top$ are formulas
- if t_1, t_2 are terms, then $t_1 \leqslant t_2$, $t_1 = t_2$ are formulas

The syntax of our theory: what it "$\phi(x, y)$"? Terms and formulas

Terms:

- every variable is a term
- $0, 1$ are terms
- if t_1, t_2 are terms, then $t_1 + t_2$ and $t_1 \cdot t_2$ are terms.

Formulas:

- $\bot, \top$ are formulas
- if t_1, t_2 are terms, then $t_1 \leqslant t_2$, $t_1 = t_2$ are formulas
- if A, B are formulas, so are $A \wedge B, A \vee B, \neg A$.

The syntax of our theory: what it "$\phi(x, y)$"? Terms and formulas

Terms:

- every variable is a term
- $0, 1$ are terms
- if t_1, t_2 are terms, then $t_1 + t_2$ and $t_1 \cdot t_2$ are terms.

Formulas:

- $\bot, \top$ are formulas
- if t_1, t_2 are terms, then $t_1 \leqslant t_2$, $t_1 = t_2$ are formulas
- if A, B are formulas, so are $A \wedge B, A \vee B, \neg A$.
- if A is a formula and x is a variable, then $\forall x A$, $\exists x A$ are formulas

The deduction system: what is "$\vdash$" in "$I\Delta_0 \vdash \phi(x, y)$"?

We use any standard deduction system for **classical**, **first-order** logic.

- disjunction introduction: $A \vdash A \vee B$ (from a proof of A you can derive a proof of $A \vee B$)

The deduction system: what is "$\vdash$" in "$I\Delta_0 \vdash \phi(x, y)$"?

We use any standard deduction system for **classical**, **first-order** logic.

- disjunction introduction: $A \vdash A \vee B$ (from a proof of A you can derive a proof of $A \vee B$)
- $\exists$ introduction: $\phi(a) \vdash \exists x, \phi(x)$ (technical restrictions on a needed)

The deduction system: what is "$\vdash$" in "$I\Delta_0 \vdash \phi(x, y)$"?

We use any standard deduction system for **classical**, **first-order** logic.

- disjunction introduction: $A \vdash A \vee B$ (from a proof of A you can derive a proof of $A \vee B$)
- $\exists$ introduction: $\phi(a) \vdash \exists x, \phi(x)$ (technical restrictions on a needed)
- double negation elimination: $\neg\neg A \vdash A$ (prove A from $\neg\neg A$)

The deduction system: what is "$\vdash$" in "$I\Delta_0 \vdash \phi(x, y)$"?

We use any standard deduction system for **classical**, **first-order** logic.

- disjunction introduction: $A \vdash A \vee B$ (from a proof of A you can derive a proof of $A \vee B$)
- $\exists$ introduction: $\phi(a) \vdash \exists x, \phi(x)$ (technical restrictions on a needed)
- double negation elimination: $\neg\neg A \vdash A$ (prove A from $\neg\neg A$)
- implication elimination (*modus ponens*): $A \to B, A \vdash B$

The deduction system: what is "$\vdash$" in "$I\Delta_0 \vdash \phi(x, y)$"?

We use any standard deduction system for **classical**, **first-order** logic.

- disjunction introduction: $A \vdash A \vee B$ (from a proof of A you can derive a proof of $A \vee B$)
- $\exists$ introduction: $\phi(a) \vdash \exists x, \phi(x)$ (technical restrictions on a needed)
- double negation elimination: $\neg\neg A \vdash A$ (prove A from $\neg\neg A$)
- implication elimination (*modus ponens*): $A \to B, A \vdash B$
- ...

The deduction system: what is "$\vdash$" in "$I\Delta_0 \vdash \phi(x, y)$"?

We use any standard deduction system for **classical**, **first-order** logic.

- disjunction introduction: $A \vdash A \vee B$ (from a proof of A you can derive a proof of $A \vee B$)
- $\exists$ introduction: $\phi(a) \vdash \exists x, \phi(x)$ (technical restrictions on a needed)
- double negation elimination: $\neg\neg A \vdash A$ (prove A from $\neg\neg A$)
- implication elimination (*modus ponens*): $A \to B, A \vdash B$
- ...

The deduction system: what is "$\vdash$" in "$I\Delta_0 \vdash \phi(x, y)$"?

We use any standard deduction system for **classical**, **first-order** logic.

- disjunction introduction: $A \vdash A \vee B$ (from a proof of A you can derive a proof of $A \vee B$)
- $\exists$ introduction: $\phi(a) \vdash \exists x, \phi(x)$ (technical restrictions on a needed)
- double negation elimination: $\neg\neg A \vdash A$ (prove A from $\neg\neg A$)
- implication elimination (*modus ponens*): $A \to B, A \vdash B$
- ...

Syntactic sugar: $A \to B := \neg A \vee B$.

The axioms: what is $I\Delta_0$? 1-BASIC axioms

Table 1: 1-BASIC axioms

Axiom	Statement
B1.	$x + 1 \neq 0$
B2.	$x + 1 = y + 1 \implies x = y$
B3.	$x + 0 = x$
B4.	$x + (y + 1) = (x + y) + 1$
B5.	$x \cdot 0 = 0$
B6.	$x \cdot (y + 1) = (x \cdot y) + x$
B7.	$(x \leq y \land y \leq x) \implies x = y$
B8.	$x \leq x + y$
C.	$0 + 1 = 1$

What can we prove so far?

Not much!

What can we prove so far?

Not much!

- Can we prove that addition is commutative?

What can we prove so far?

Not much!

- Can we prove that addition is commutative? NO!

What can we prove so far?

Not much!

- Can we prove that addition is commutative? NO!
- Can we prove that addition is associative?

What can we prove so far?

Not much!

- Can we prove that addition is commutative? NO!
- Can we prove that addition is associative? NO!

Axiom schema of induction

Definition (Induction Scheme).
If Φ is a set of formulas, then Φ-IND axioms are the formulas

$$\left(\varphi(0) \ \wedge \ \forall x \, (\varphi(x) \to \varphi(x+1))\right) \ \to \ \forall z \, \varphi(z),$$

where $\varphi \in \Phi$. $\varphi(x)$ may have free variables other than x.

Axiom schema of induction

Definition (Induction Scheme).
If Φ is a set of formulas, then Φ-IND axioms are the formulas

$$(\varphi(0) \;\wedge\; \forall x\,(\varphi(x) \to \varphi(x+1))) \;\to\; \forall z\,\varphi(z),$$

where $\varphi \in \Phi$. $\varphi(x)$ may have free variables other than x.

The theory having axioms **B1-B8**, together with induction for arbitrary formulas from our vocabulary, *is* the **Peano** arithmetic (a very strong system).

Axiom schema of induction

Definition (Induction Scheme).
If Φ is a set of formulas, then Φ-IND axioms are the formulas

$$(\varphi(0) \ \wedge \ \forall x\,(\varphi(x) \to \varphi(x+1))) \ \to \ \forall z\,\varphi(z),$$

where $\varphi \in \Phi$. $\varphi(x)$ may have free variables other than x.

The theory having axioms **B1**-**B8**, together with induction for arbitrary formulas from our vocabulary, *is* the **Peano** arithmetic (a very strong system).

By carefully controling Φ, we obtain **interesting** theories.

Complexity of formulas

Definition (Bounded Quantifiers).

$$\exists x \leq t\, A \;:=\; \exists x\, (x \leq t \wedge A)$$

$$\forall x \leq t\, A \;:=\; \forall x\, (x \leq t \rightarrow A)$$

(the variable x must not occur in the term t)
Quantifier that occur in this form are **bounded**.

Complexity of formulas

Definition (Bounded Quantifiers).

$$\exists x \leq t\, A \;:=\; \exists x\, (x \leq t \wedge A)$$

$$\forall x \leq t\, A \;:=\; \forall x\, (x \leq t \rightarrow A)$$

(the variable x must not occur in the term t)
Quantifier that occur in this form are **bounded**.

A formula is Δ_0 (**bounded**) if every quantifier in it is bounded.
A formula is Σ_1 if it is of the form $\exists x_1, \ldots, \exists x_k \phi$ and ϕ is bounded.

Case 2: $\Phi = \Delta_0$

1-BASIC axioms together with induction for bounded formulas only give us a well-studied system called $I\Delta_0$.

The following formulas (and their universal closures) are theorems of $I\Delta_0$ (Cook & Nguyen, 2010):

- $x + y = y + x$ *(commutativity of* $+$*)*

Case 2: $\Phi = \Delta_0$

1-BASIC axioms together with induction for bounded formulas only give us a well-studied system called $I\Delta_0$.

The following formulas (and their universal closures) are theorems of $I\Delta_0$ (Cook & Nguyen, 2010):

- $x + y = y + x$ *(commutativity of* $+$*)*
- $(x + y) + z = x + (y + z)$ *(associativity of* $+$*)*

Case 2: $\Phi = \Delta_0$

1-BASIC axioms together with induction for bounded formulas only give us a well-studied system called $I\Delta_0$.

The following formulas (and their universal closures) are theorems of $I\Delta_0$ (Cook & Nguyen, 2010):

- $x + y = y + x$ *(commutativity of $+$)*
- $(x + y) + z = x + (y + z)$ *(associativity of $+$)*
- $x \leq x$

Case 2: $\Phi = \Delta_0$

1-BASIC axioms together with induction for bounded formulas only give us a well-studied system called $I\Delta_0$.

The following formulas (and their universal closures) are theorems of $I\Delta_0$ (Cook & Nguyen, 2010):

- $x + y = y + x$ *(commutativity of +)*
- $(x + y) + z = x + (y + z)$ *(associativity of +)*
- $x \leq x$
- $0 \leq x$

Case 2: $\Phi = \Delta_0$

1-BASIC axioms together with induction for bounded formulas only give us a well-studied system called $I\Delta_0$.

The following formulas (and their universal closures) are theorems of $I\Delta_0$ (Cook & Nguyen, 2010):

- $x + y = y + x$ *(commutativity of +)*
- $(x + y) + z = x + (y + z)$ *(associativity of +)*
- $x \leq x$
- $0 \leq x$
- $\forall x\, \forall y\, \big(0 < x \to \exists q\, \exists r\, (\, r < x \wedge y = x \cdot q + r\,)\big)$ *(division theorem)*

Defining new functions in $I\Delta_0$

We say that a function $f(\vec{x})$ is *provably total* in $I\Delta_0$ if there is a formula $\phi(\vec{x}, y)$ in Σ_1 (i.e. of the form $\exists \ldots \exists \psi$ for ψ bounded) such that:

$$I\Delta_0 \vdash \forall x \exists ! y \phi(\vec{x}, y)$$

and that

$$y = f(\vec{x}) \iff \phi(\vec{x}, y)$$

What functions can $I\Delta_0$ *define* and what it can't?

Examples:

What functions can $I\Delta_0$ *define* and what it can't?

Examples:

- the function $LimitedSub(x, y) := \max\{0, x - y\}$

What functions can $I\Delta_0$ *define* and what it can't?

Examples:

- the function $\mathit{LimitedSub}(x, y) := \max\{0, x - y\}$
- the function x div $y := \lfloor x/y \rfloor$ is defined by

$$z = \lfloor x/y \rfloor \ \leftrightarrow \ ((y \cdot z \leq x \wedge x < y(z+1)) \vee (y = 0 \wedge z = 0)).$$

What functions can $I\Delta_0$ *define* and what it can't?

Examples:

- the function $\mathit{LimitedSub}(x, y) := \max\{0, x - y\}$
- the function x div $y := \lfloor x/y \rfloor$ is defined by

$$z = \lfloor x/y \rfloor \leftrightarrow ((y \cdot z \leq x \wedge x < y(z+1)) \vee (y = 0 \wedge z = 0)).$$

- x mod y

What functions can $I\Delta_0$ *define* and what it can't?

Examples:

- the function $\mathit{LimitedSub}(x, y) := \max\{0, x - y\}$
- the function $x \text{ div } y := \lfloor x/y \rfloor$ is defined by

$$z = \lfloor x/y \rfloor \;\leftrightarrow\; ((y \cdot z \leq x \wedge x < y(z+1)) \vee (y = 0 \wedge z = 0)).$$

- $x \bmod y$
- $\lfloor \sqrt{x} \rfloor$

What functions can $I\Delta_0$ *define* and what it can't?

Examples:

- the function $\mathit{LimitedSub}(x, y) := \max\{0, x - y\}$
- the function $x \text{ div } y := \lfloor x/y \rfloor$ is defined by

$$z = \lfloor x/y \rfloor \ \leftrightarrow\ ((y \cdot z \leq x \wedge x < y(z+1)) \vee (y = 0 \wedge z = 0)).$$

- $x \bmod y$
- $\lfloor \sqrt{x} \rfloor$
- ...

What functions can $I\Delta_0$ *define* and what it can't?

Examples:

- the function $\mathit{LimitedSub}(x, y) := \max\{0, x - y\}$
- the function $x \text{ div } y := \lfloor x/y \rfloor$ is defined by

$$z = \lfloor x/y \rfloor \leftrightarrow ((y \cdot z \leq x \wedge x < y(z + 1)) \vee (y = 0 \wedge z = 0)).$$

- $x \bmod y$
- $\lfloor \sqrt{x} \rfloor$
- ...

What functions can $I\Delta_0$ *define* and what it can't?

Examples:

- the function $\mathit{LimitedSub}(x, y) := \max\{0, x - y\}$
- the function $x \text{ div } y := \lfloor x/y \rfloor$ is defined by

$$z = \lfloor x/y \rfloor \;\leftrightarrow\; ((y \cdot z \leq x \wedge x < y(z+1)) \vee (y = 0 \wedge z = 0)).$$

- $x \bmod y$
- $\lfloor \sqrt{x} \rfloor$
- ...

For all of these, you need to prove existence and uniqueness of the result.

What functions can $I\Delta_0$ *define* and what it can't?

Examples:

- the function $\mathit{LimitedSub}(x, y) := \max\{0, x - y\}$
- the function $x \operatorname{div} y := \lfloor x/y \rfloor$ is defined by

$$z = \lfloor x/y \rfloor \ \leftrightarrow\ ((y \cdot z \leq x \wedge x < y(z+1)) \vee (y = 0 \wedge z = 0)).$$

- $x \bmod y$
- $\lfloor \sqrt{x} \rfloor$
- ...

For all of these, you need to prove existence and uniqueness of the result.

BUT: $I\Delta_0$ can't "prove total" the exponential function $(x \mapsto 2^x)$!

What functions can $I\Delta_0$ *define* and what it can't?

Examples:

- the function $\mathit{LimitedSub}(x, y) := \max\{0, x - y\}$
- the function $x \text{ div } y := \lfloor x/y \rfloor$ is defined by

$$z = \lfloor x/y \rfloor \;\leftrightarrow\; ((y \cdot z \leq x \wedge x < y(z+1)) \vee (y = 0 \wedge z = 0)).$$

- $x \text{ mod } y$
- $\lfloor \sqrt{x} \rfloor$
- ...

For all of these, you need to prove existence and uniqueness of the result.

BUT: $I\Delta_0$ can't "prove total" the exponential function $(x \mapsto 2^x)$!

NOTE: the computational content of $I\Delta_0$ is well-studied.
NOTE: $I\Delta_0$ doesn't align well with practical computer science.

Theories corresponding to complexity classes

The idea is similar.

Theories corresponding to complexity classes

The idea is similar.

- We still operate in first-order, classical logic.

Theory	Characterizes	Examples
V^0	FAC^0	$\nvdash$ Pigeonhole; $\vdash$ properties of binary $+$
VTC^0	FTC^0	$\vdash$ Pigeonhole; defines sorting
VL	FLOGSPACE	$\cdots$
V^1	FPTIME	$\cdots$

If time allows, we will get back to details of the definition.

Theories corresponding to complexity classes

The idea is similar.

- We still operate in first-order, classical logic.
- Instead of one sort, we have two:

Theory	Characterizes	Examples
V^0	FAC^0	$\nvdash$ Pigeonhole; $\vdash$ properties of binary $+$
VTC^0	FTC^0	$\vdash$ Pigeonhole; defines sorting
VL	FLOGSPACE	$\cdots$
V^1	FPTIME	$\cdots$

If time allows, we will get back to details of the definition.

Theories corresponding to complexity classes

The idea is similar.

- We still operate in first-order, classical logic.
- Instead of one sort, we have two:
  - `num` (representing unary numbers)

Theory	Characterizes	Examples
V^0	FAC^0	$\nvdash$ Pigeonhole; $\vdash$ properties of binary $+$
VTC^0	FTC^0	$\vdash$ Pigeonhole; defines sorting
VL	FLOGSPACE	$\cdots$
V^1	FPTIME	$\cdots$

If time allows, we will get back to details of the definition.

Theories corresponding to complexity classes

The idea is similar.

- We still operate in first-order, classical logic.
- Instead of one sort, we have two:
  - `num` (representing unary numbers)
  - `str` (representing binary strings)

Theory	Characterizes	Examples
V^0	FAC^0	$\nvdash$ Pigeonhole; $\vdash$ properties of binary $+$
VTC^0	FTC^0	$\vdash$ Pigeonhole; defines sorting
VL	FLOGSPACE	$\cdots$
V^1	FPTIME	$\cdots$

If time allows, we will get back to details of the definition.

Theories corresponding to complexity classes

The idea is similar.

- We still operate in first-order, classical logic.
- Instead of one sort, we have two:
  - `num` (representing unary numbers)
  - `str` (representing binary strings)
- Instead of induction we have finite set comprehension (finite sets $\equiv$ binary strings)

Theory	Characterizes	Examples
V^0	FAC^0	$\nvdash$ Pigeonhole; $\vdash$ properties of binary $+$
VTC^0	FTC^0	$\vdash$ Pigeonhole; defines sorting
VL	FLOGSPACE	$\cdots$
V^1	FPTIME	$\cdots$

If time allows, we will get back to details of the definition.

How would you even formalize this field?

Requirements on the product

Transfer proofs of the form $V^0 \vdash x + y = y + x$ from paper to computer.

Make "cheating" difficult or visible for the reader.

Enable easy interactive proving inside of the weak arithmetic.

Problems to avoid

No way to express that *Prop* is Δ_0 or Σ_1 in any of the existing systems. Rocq, Lean and Isabelle/Pure all don't foster a shallow embedding.

```
inductive Formula
| false : Formula
| eq (term1 term2 : Term) : Formula
| implies (f1 f2 : Formula) : Formula
```

Defining a proof system from scratch can take years of works to become usable

The 90% solution

```
class IOPENModel (M : Type _) where
  num : Type*
  B1 : num.realizes B1_statement
  B2 : num.realizes B2_statement
  open_induction (phi: Formula) :
    phi.IsOpen -> num.realizes (makeInduction phi)

theorem add_assoc (M : IOPENModel)
  : forall x y z : M.num, (x + y) + z = x + (y + z) := by
  have ind := M.open_induction $
    ((x' +' y') + z') =' (x' + (y' + z'))
  -- simps of axioms
  intro x y z
```

Another design will be necessary for proof-theoretical results

What's formalized?

- the $I\Delta_0$ theory and the two-sorted V^0 theory
- basic properties of the $I\Delta_0$ system
  proofs by induction on a Δ_0 formula
- partial proof of $V^0 \vdash MIN$, first step towards obtaining induction in V^0

How it looks like?

```
intro x
  apply ind
  · intro a ha ha'
    exists a
    constructor
    · apply b8
    · rfl
```

Thanks!

https://github.com/ruplet/formalization-of-bounded-arithmetic

This project has been supported by the ZSM IDUB program at the University of Warsaw

Bonus: finite axiomatizability of V^0

The theory V^0 is finitely axiomatizable(Cook & Nguyen, 2010).

You don't need an induction axiom scheme, nor a comprehension axiom scheme. The instantiations of induction to around 20 formulas and of comprehension to 12 formulas suffice.

Moreover, since the theories VC expressing complexity classes C are constructed from axioms of V^0 + a complete problem for C taken as an axiom. So they are also finitely axiomatizable and (very) expressive.

Perhaps V^0 is a good theory for automated proof search. I haven't managed to explore this direction yet.

V^0 definition: 2-BASIC axioms

Two sorts: unary numbers $(x, y, z, \ldots)$, binary strings $(X, Y, Z, \ldots)$.

Symbols: $L_A^2 = [0, 1, +, \cdot, \text{len}, =_{num}, =_{str}, \leq, \in]$.

B1. $x + 1 \neq 0$

B2. $x + 1 = y + 1 \to x = y$

B3. $x + 0 = x$

B4. $x + (y + 1) = (x + y) + 1$

B5. $x \cdot 0 = 0$

B6. $x \cdot (y + 1) = (x \cdot y) + x$

B7. $(x \leq y \land y \leq x) \to x = y$

B8. $x \leq x + y$

B9. $0 \leq x$

B10. $x \leq y \lor y \leq x$

B11. $x \leq y \leftrightarrow x < y + 1$

B12. $x \neq 0 \to \exists y \leq x\, (y + 1 = x)$

L1. $X(y) \to y < |X|$

L2. $y + 1 = |X| \to X(y)$

SE. $(|X| = |Y| \land \forall i < |X|\, (X(i) \leftrightarrow Y(i))) \to X = Y$

Notation: $\exists X \leqslant y \phi := \exists X(|X| \leqslant y \wedge \phi)$.

Definition (Comprehension Axiom).
If Φ is a set of formulas, the comprehension scheme for Φ (denoted Φ-COMP) consists of all instances

$$\exists X \leq y \, \forall z < y \, (X(z) \leftrightarrow \varphi(z)),$$

where $\varphi(z) \in \Phi$ and X does not occur free in $\varphi(z)$.
Here $\varphi(z)$ may have additional free variables of either sort besides z.

Definition (V_i).
For $i \geq 0$, the theory V_i has vocabulary L_A^2 and is axiomatized by **2-BASIC** together with Σ_i^B-COMP.

Bibliography

- Jiatu Li's introduction from 1st July 2025: https://eccc.weizmann.ac.il/report/2025/086/

Cook, S., & Nguyen, P. (2010). *Logical foundations of proof complexity*. Cambridge University Press. https://doi.org/10.1017/CBO9780511676277http://web.archive.org/web/20240713034207/https://www.karlin.mff.cuni.cz/~krajicek/cook-nguyen.pdf