Replacing Induction by Inductive Definitions in Bounded Arithmetic

Naohi Eguchi
Mathematical Institute, Tohoku University, Japan

supported by the JTF grant Philosophical frontiers in Reverse Mathematics
Introduction

\[ f \text{ is computable} \iff \exists \text{ program to compute } f \]
Introduction

$f$ is computable $\iff \exists$ program to compute $f$

$f \in \text{FP} \iff \exists$ program to compute $f(x)$ within a $p(|x|)$ step

($f$ is polytime computable)
Introduction

\[ f \text{ is computable } \iff \exists \text{ program to compute } f \]
\[ f \in \text{FP } \iff \exists \text{ program to compute } f(x) \text{ within a } p(|x|) \text{ step} \]

Formalise!

- to find new aspects of computations in complexity classes like \( \mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} \subseteq \mathbf{EXP} \).
- to understand how difficult to solve \( \mathbf{P} \neq \mathbf{NP} \), \( \mathbf{P} \neq \mathbf{PSPACE} \), \( \mathbf{PSPACE} \neq \mathbf{EXP} \)?
Introduction

\[ f \text{ is computable } \iff \exists \text{ program to compute } f \]
\[ f \in \text{FP } \iff \exists \text{ program to compute } f(x) \text{ within a } p(|x|) \text{ step } \]

(f is polytime computable)

Formalise!

- to find new aspects of computations in complexity classes like \( P \subseteq NP \subseteq PSPACE \subseteq EXP \).
- to understand how difficult to solve \( P \neq NP \), \( P \neq PSPACE \), \( PSPACE \neq EXP \)?

\( \implies \text{Bounded arithmetic!} \)
<table>
<thead>
<tr>
<th>Class</th>
<th>Condition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f \in \text{FP}$</td>
<td>$\Leftrightarrow \quad S_2^1 \vdash \exists \text{ program to compute } f(x)$ within a $p(</td>
<td>x</td>
</tr>
<tr>
<td>$f \in \text{FP}^{\text{NP}}$</td>
<td>$\Leftrightarrow \quad S_2^2 \vdash \exists \text{ program to compute } f(x) \ldots$ with $\text{NP}$-oracles</td>
<td></td>
</tr>
<tr>
<td>$f \in \text{FP}^{\Sigma_n^p}$</td>
<td>$\Leftrightarrow \quad S_2^{n+1} \vdash \exists \text{ program to compute } f(x) \ldots$ with $\Sigma_n^p$-oracles</td>
<td></td>
</tr>
<tr>
<td>$f \in \text{FPSPACE}$</td>
<td>$\Leftrightarrow \quad U_1^1 \vdash \exists \text{ program to compute } f(x)$ with a $p(</td>
<td>x</td>
</tr>
</tbody>
</table>
Overview 2/4

\[ f \in \text{FP} \iff V^1 \vdash \exists \text{ program to compute } f(x) \text{ within a } p(|x|) \text{ step} \]

\[ f \in \text{FP}^\Sigma_n \iff V^{n+1} \vdash \exists \text{ program to compute } f(x) \ldots \text{ with } \Sigma_n^P \text{-oracles} \]

(D. Zambella ’96)

\[ f \in \text{FPSPACE} \iff W^1_1 \vdash \exists \text{ program to compute } f(x) \text{ with a } p(|x|)\text{-space} \]

(A. Skelley ’05)

\[ S^2_n, U^1_1, V^n, W^1_1 \text{ are contained in } \mathbf{I\Delta}_0(\text{exp}). \]
Overview 3/4

$S^2_n, U^1_1, V^n, W^1_1$ are based on either induction or (bit-)comprehension.

Motivation 1/2. In finite model theory

$f \in FP \iff \text{LFP} \models \exists \text{ program to compute } f(x) \text{ within a } p(|x|) \text{ step (witnessed by the least fixed point of a monotone operator)}$
Overview 4/4

$S_2^n, U_1^1, V^n, W_1^1$ are based on either induction or (bit-)comprehension.

Motivation 2/2. in finite model theory

\[ f \in \Leftrightarrow \text{PFP} \models \exists \text{ program to compute } f(x) \text{ within a } p(|x|) \text{ step (witnessed by a fixed point of a non-monotone operator)} \]

This work (has just started): Can induction (or comprehension) be replaced by inductive definitions?
1. Formalising computations.
2. Foundations of 2nd order bounded arithmetic.
3. Fixed point axioms in bounded arithmetic.
4. Constructing polytime configurations.
5. Results.
6. Further directions (PSPACE).
7. Conclusion
Formalising computations 1/2

$f$ is computable $\iff \exists \text{ program to compute } f$

This gives rise to:

**Def** Let $\Phi$: a set of formulas $\subseteq \Sigma^0_1$ & $f$: a function. $f$ is $\Phi$-definable in $T$ if $\exists A(\vec{x}, y) \in \Phi$ such that

1. All free variables in $A(\vec{x}, y)$ are indicated.
2. $n = f(\vec{m}) \iff \mathbb{N} \models A(\vec{m}, n)$ for $\forall \vec{m}, n \in \mathbb{N}$.
3. $T \vdash \forall \vec{x} \exists! y A(\vec{x}, y)$.
Classical facts:

1. \( f: \) primitive recursive \( \iff \) \( f: \Sigma^0_1\)-definable in \( I\Sigma_1 \).
   (Parsons ’70, Mints ’73, Buss ’86 and Takeuti ’87)

2. \( f \in \text{FP} \iff f: \Sigma^b_1\)-definable in \( S^1_2 \).
   (Buss ’86)
   • The start of machine-independent (proof-theoretic) characterisations of complexity classes.

These are not discussed about in this talk…
Contents

1. Formalising computations.
2. Foundations of 2nd order bounded arithmetic.
3. Fixed point axioms in bounded arithmetic.
4. Constructing polytime configurations.
5. Results.
6. Further directions (PSPACE).
7. Conclusion
Languages of 2nd order bounded arithmetic:
1. $0, S, +$ and $\cdot$.
2. $\lfloor \frac{x}{2} \rfloor$, $|x| = \lceil \log_2(x + 1) \rceil$ and $|X|$.

Importantly $x \# y = 2^{|x| \cdot |y|}$ is not included.

Intuition:
1. $X, Y, Z \cdots \in \mathbb{N} \{0, 1\}$.
2. $|X| = l$ if $X \equiv i_0i_1\cdots i_{l-1}$ & $i_j \in \{0, 1\}$.
3. $j \in X \iff i_j = 1$ if $X \equiv i_0i_1\cdots i_{l-1}$. 
Def \((\Sigma^B_1\)-formulas)\)

1. \(\Sigma^B_0 = \Pi^B_0\): the set of formulas containing only bounded number quantifiers \(\exists x \leq t\).
2. \(\exists \vec{X} (|\vec{X}| \leq \vec{t} \land \varphi(\vec{X})) \in \Sigma^B_{n+1}\) if \(\varphi \in \Pi^B_n\).
Foundations of 2nd order bounded arithmetic 2/3

Def (\(\Sigma^B_1\)-formulas)
1. \(\Sigma^B_0 = \Pi^B_0\): the set of formulas containing only bounded number quantifiers \(\exists x \leq t\).
2. \(\exists \vec{X} (|\vec{X}| \leq \vec{t} \land \varphi(\vec{X})) \in \Sigma^B_{n+1}\) if \(\varphi \in \Pi^B_n\).

Def (Bit-comprehension axiom)
\(\exists X\) s.t. \(|X| \leq y \land \forall x < y (x \in X \leftrightarrow \varphi(x))\)

\(\Sigma^B_n\)-COMP: CA with \(\varphi\) restricted to \(\Sigma^B_n\).

Note: \(\bigcup_{n \in \mathbb{N}} \Sigma^B_n \subseteq \Delta^0_1(\exp) \subseteq \Sigma^0_1\) by definition.
<table>
<thead>
<tr>
<th></th>
<th>2nd order arith.</th>
<th>2nd order BA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st order objects</td>
<td>elements of $\mathbb{N}$</td>
<td>$\leq p(</td>
</tr>
<tr>
<td>2nd order objects</td>
<td>$f : \mathbb{N} \rightarrow \mathbb{N}$</td>
<td>$f : p(</td>
</tr>
<tr>
<td>typical classes of formulas</td>
<td>$\Sigma_n^1$</td>
<td>$\Sigma_n^B$</td>
</tr>
</tbody>
</table>

($p$: polynomial)
## Foundations of 2nd order bounded arithmetic 3/3

<table>
<thead>
<tr>
<th></th>
<th>2nd order arith.</th>
<th>2nd order BA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st order objects</td>
<td>elements of $\mathbb{N}$</td>
<td>$\leq p(</td>
</tr>
<tr>
<td>2nd order objects</td>
<td>$f : \mathbb{N} \rightarrow \mathbb{N}$</td>
<td>$f : p(</td>
</tr>
</tbody>
</table>
| typical classes of formulas | $\Sigma^1_n$ | $\Sigma^B_n$ | *(p: polynomial)*

**Def** \( V^n := \text{BASIC} + \Sigma^B_n\)-COMP.

(\text{BASIC}: defining axioms for the function symbols)

**Thm** (Zambella '96)

\( f \in \text{FP}^{\Sigma^p_n} \iff f : \Sigma^B_{n+1}\)-definable in \( V^{n+1} \).
1. Formalising computations.
2. Foundations of 2nd order bounded arithmetic.
3. Fixed point axioms in bounded arithmetic.
4. Constructing polytime configurations.
5. Results.
6. Further directions (PSPACE).
7. Conclusion
Fixed point axioms in bounded arithmetic

We introduce:

Def (Least fixed point axiom) If $\forall X \subseteq Y \varphi(x, X) \rightarrow \varphi(x, Y)$, then $\exists X$ s.t. $|X| \leq y$,

- $\forall x < y [\varphi(x, X) \rightarrow x \in X]$ (closedness),
- $\forall Y [|Y| \leq y' \land \forall x < y' (\varphi(x, Y) \rightarrow x \in Y) \rightarrow X \subseteq Y]$ (leastness).

Def $\Sigma^B_n$-LFP: LFPA with $\varphi$ restricted to $\Sigma^B_n$. 
We introduce:

**Def (Least fixed point axiom)** If $\forall X \subseteq Y \varphi(x, X) \rightarrow \varphi(x, Y)$, then $\exists X$ s.t. $|X| \leq y$,

- $\forall x < y[\varphi(x, X) \rightarrow x \in X]$ (closedness),
- $\forall Y[|Y| \leq y' \land \forall x < y'(\varphi(x, Y) \rightarrow x \in Y) \rightarrow X \subseteq Y]$ (leastness).

**Def $\Sigma^B_n$-LFP:** LFPA with $\varphi$ restricted to $\Sigma^B_n$.

We show:

$f \in FP \iff f: \Sigma^B_1$-definable in $V^0 + \Sigma^B_0$-LFP.
1. Formalising computations.
2. Foundations of 2nd order bounded arithmetic.
3. Fixed point axioms in bounded arithmetic.
4. Constructing polytime configurations.
5. Results.
6. Further directions (PSPACE).
7. Conclusion
Constructing polytime configurations 1/3

\[ f \in \text{FP} \iff \exists \text{ Turing-program to compute } f(x) \text{ within a } p(|x|) \text{ step} \]
$f \in \text{FP} \iff \exists \text{Turing-program to compute } f(x) \text{ within a } p(|x|) \text{ step}$

$\iff \exists C_0, \ldots, C_{p(|x|)-1}: \text{Turing-configurations leading to } f(x)$

Defined via induction:

$\hat{C}_0 = C_0$ & $\hat{C}_{n+1} = \hat{C}_n \hat{\sim} C_{n+1}$ ($\hat{\sim}$: concatenation)
Constructing polytime configurations 1/3

\[ f \in \text{FP} \iff \exists \text{Turing-program to compute } f(x) \text{ within a } p(|x|) \text{ step} \]

\[ \iff \exists C_0, \ldots, C_{p(|x|)-1}: \text{Turing-configurations leading to } f(x) \]

Defined via induction:
\[ \hat{C}_0 = C_0 \land \hat{C}_{n+1} = \hat{C}_n \tilde{\triangleright} C_{n+1} \quad (\tilde{\triangleright}: \text{concatenation}) \]

Fact \[ f \in \text{FP} \iff f: \Sigma_1^B\text{-definable in } V^1 \]

This fact follows from:
Lem \[ V^0 + \Sigma_n^B\text{-COMP} =_{\Sigma_n^B} V^0 + \Sigma_n^B\text{-IND}. \]
Replace induction by inductive definitions:
\( \tilde{C}_0 := C_0 \) & \( \tilde{C}_{n+1} := \tilde{C}_n \sim C_{n+1} \).

Then \( \lim_{n<\omega} \tilde{C}_n = \hat{C}_p(|x|) - 1 \).
Replace induction by inductive definitions:
\[ \tilde{C}_0 := C_0 & \tilde{C}_{n+1} := \tilde{C}_n \upharpoonright C_{n+1}. \]

Then \( \lim_{n<\omega} \tilde{C}_n = \hat{C}_p(|x|) - 1. \)

Formalise this construction:
1. \( |C_n| \leq q(|x|) \) for some polynomial \( q. \)
2. \( C_{\text{init}}(x) \): initial configuration on input \( x. \)
3. \( C_{\text{next}}(x, X) \): next configuration of \( X \) on input \( x. \)

Then \( y \in C_{\text{init}}(x), y \in C_{\text{next}}(x, X) : \Sigma_0^B \)
Define a formula $\varphi(y, x, X)$ expressing:

- $y \in C_{\text{initial}}(x)$ or
- $y \in X \lor y \in C_{\text{next}}(x, \text{last}(q(|x|), X))$

where $\text{last}(j, X)$ denotes the last $j$ bits of $X$. 
Define a formula $\varphi(y, x, X)$ expressing:

- $y \in C_{\text{initial}}(x)$ or
- $y \in X \lor y \in C_{\text{next}}(x, \text{last}(q(|x|), X))$

where $\text{last}(j, X)$ denotes the last $j$ bits of $X$.

Then $\forall X \subseteq Y[\varphi(y, x, X) \rightarrow \varphi(y, x, Y)]$.

Hence $\lim_{n<\omega} \tilde{C}_n = \tilde{C}_p(|x|) - 1$ is the least fixed point of the monotone operator defined by $\varphi$. 
Contents

1. Formalising computations.
2. Foundations of 2nd order bounded arithmetic.
3. Fixed point axioms in bounded arithmetic.
4. Constructing polytime configurations.
5. Results.
6. Further directions (PSPACE).
7. Conclusion
Thm

\[ f \in \text{FP} \iff f: \Sigma^B_1\text{-definable in } V^0 + \Sigma^B_0\text{-LFP.} \]

(\Rightarrow): already shown (informally)

(\Leftarrow): \[ V^0 \vdash \Sigma^B_{n+1}\text{-COMP} \rightarrow \Sigma^B_n\text{-LFP} \]
(Compare: \( \text{ACA}_0 \vdash \Sigma^1_{n+1}\text{-CA} \rightarrow \Sigma^1_n\text{-LFP} \))

\[ f: \Sigma^B_1\text{-definable in } V^0 + \Sigma^B_0\text{-LFP} \]
\[ \Rightarrow f: \Sigma^B_1\text{-definable in } V^1 \]
\[ \Rightarrow f \in \text{FP} \quad \text{(Zambella ’96)} \]
Thm $f \in \text{FP} \iff f: \Sigma^B_1$-definable in $\mathcal{V}^0 + \Sigma^B_0$-LFP.

Seems to be possible to generalise to:

Thm $f \in \text{FP}^\Sigma^n \iff f: \Sigma^B_{n+1}$-definable in $\mathcal{V}^0 + \Sigma^B_n$-LFP.

(Should be checked carefully!)

Recall:

Fact $f \in \text{FP}^\Sigma^n \iff f: \Sigma^B_{n+1}$-definable in $\mathcal{V}^{n+1}$. 
1. Formalising computations.
2. Foundations of 2nd order bounded arithmetic.
3. Fixed point axioms in bounded arithmetic.
4. Constructing polytime configurations.
5. Results.
6. Further directions (PSPACE).
7. Conclusion
Further directions (PSPACE) 1/2

The previous inductive definition of $\tilde{C}_n$ does not work for PSPACE.

$$f \in \text{FPSPACE} \iff \exists \text{Turing-program to compute } f(x) \text{ with a } p(|x|) \text{ space}$$
Further directions (PSPACE) 1/2

The previous inductive definition of $\tilde{C}_n$ does not work for PSPACE.

$$f \in \text{FPSPACE} \iff \exists \text{Turing-program to compute } f(x) \text{ with a } p(|x|) \text{ space}$$

$$\iff \exists C_0, \ldots, C_{2O(p(|x|))} : \text{Turing-configurations leading to } f(x)$$

Why? $$2^{O(p(|x|))} \leq |C_0 \cdots \cdots \cdots C_{2O(p(|x|))}|$$
Partial fixed points.

Define a non-monotone $\Sigma^B_0$-formula $\varphi(y, x, X)$:

- $y \in C_{\text{init}}(x)$ or
- $y \in X \lor y \in C_{\text{next}}(x, X)$.

The final configuration $C_{2^{O(p(|x|))}}$ is a fixed point of $\varphi$ but may not be the least one.

**Problem:** Any natural formulation of partial fixed point principles?
Conclusion

- Formalising construction of witnessing configurations for computable functions to understand what is the most essential in their computations.
- $\Sigma_{n+1}^B$-comprehension, $\Sigma_{n+1}^B$-induction can be replaced by $\Sigma_n^B$-least fixed point axiom.
- FP is captured by $\Sigma_0^B$-LFP (over $V^0$).
- Is FPSPACE captured by $\Sigma_0^B$-PFP?
- This approach may shed light on a big open problem $P \neq \text{PSPACE}$?