Enumerating Error Bounded Polytime Algorithms Through Arithmetical Theories

Melissa Antonelli **Ugo Dal Lago** **Davide Davoli**
Isabel Oitavem **Paolo Pistone**

Structure Meets Power, Boston, *June 25th 2023*

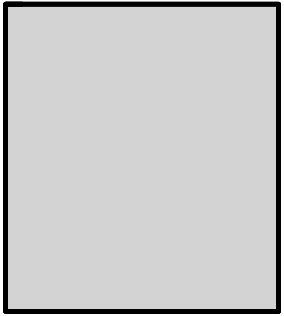
Part I

Context and Motivations

Implicit Computational Complexity

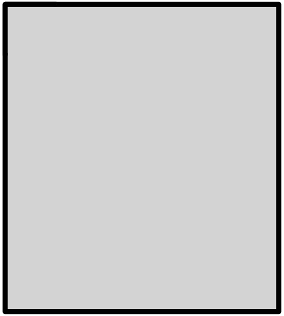
Programs

$\{0,1\}^*$

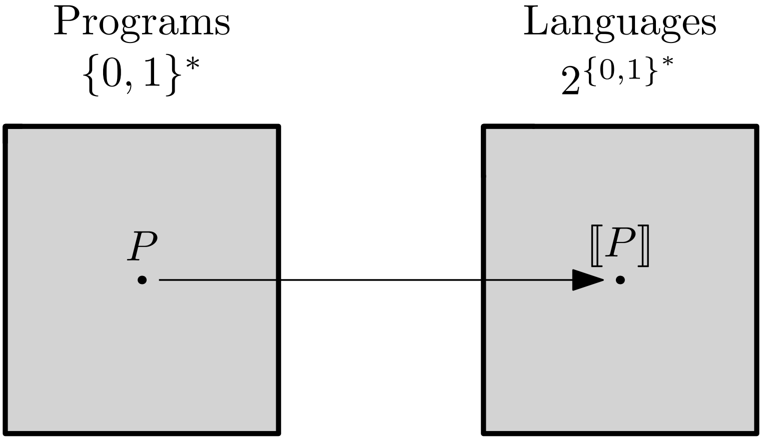


Languages

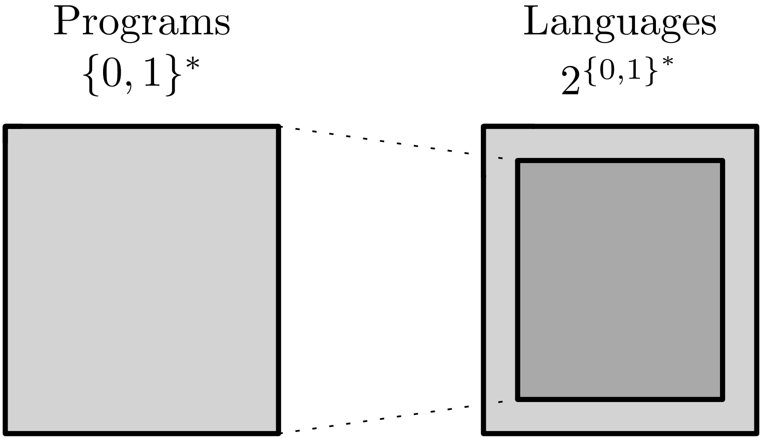
$2^{\{0,1\}^*}$



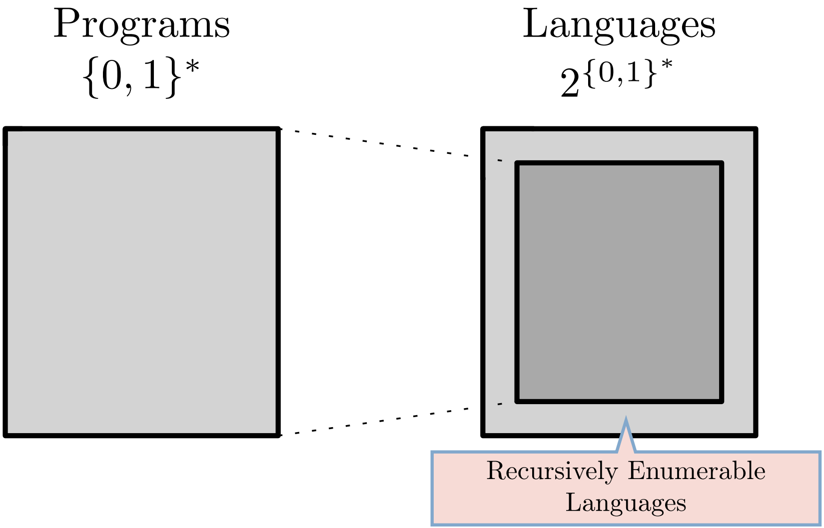
Implicit Computational Complexity



Implicit Computational Complexity



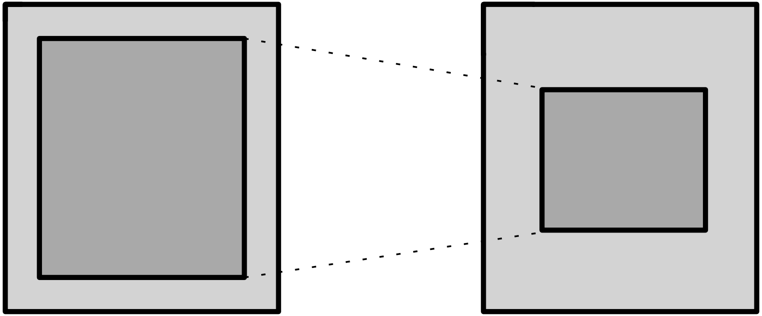
Implicit Computational Complexity



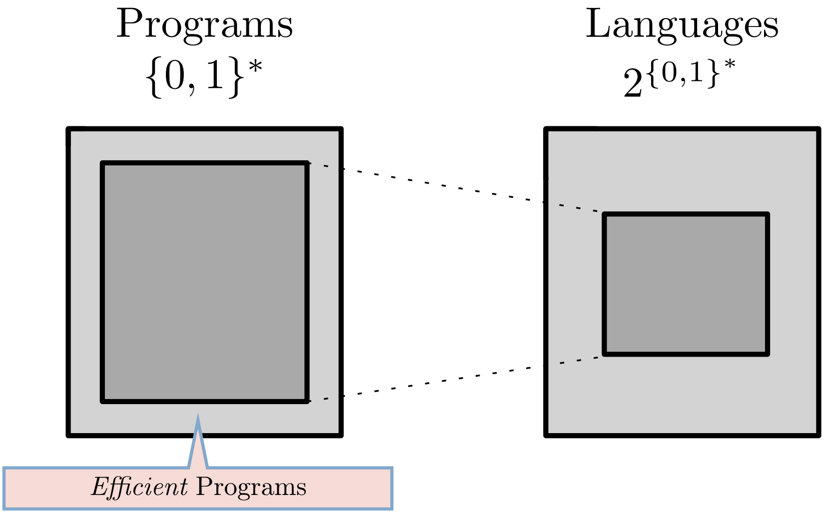
Implicit Computational Complexity

Programs
$\{0,1\}^*$

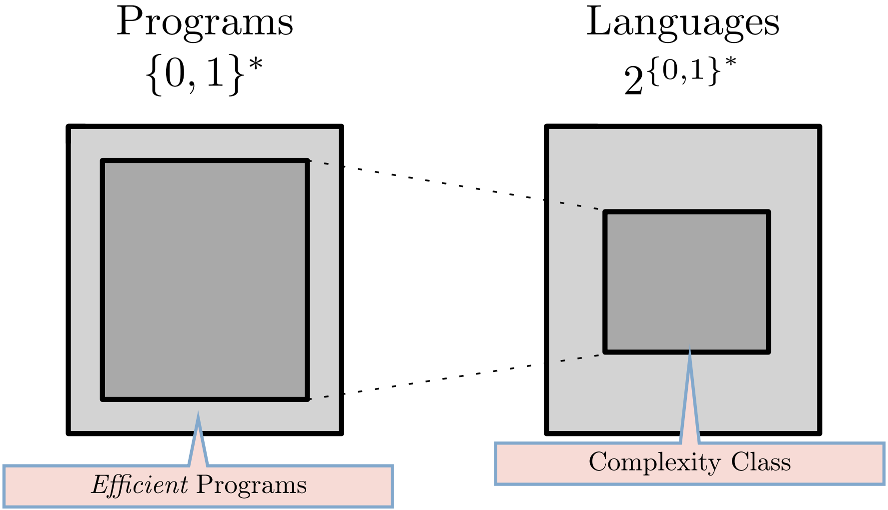
Languages
$2^{\{0,1\}^*}$



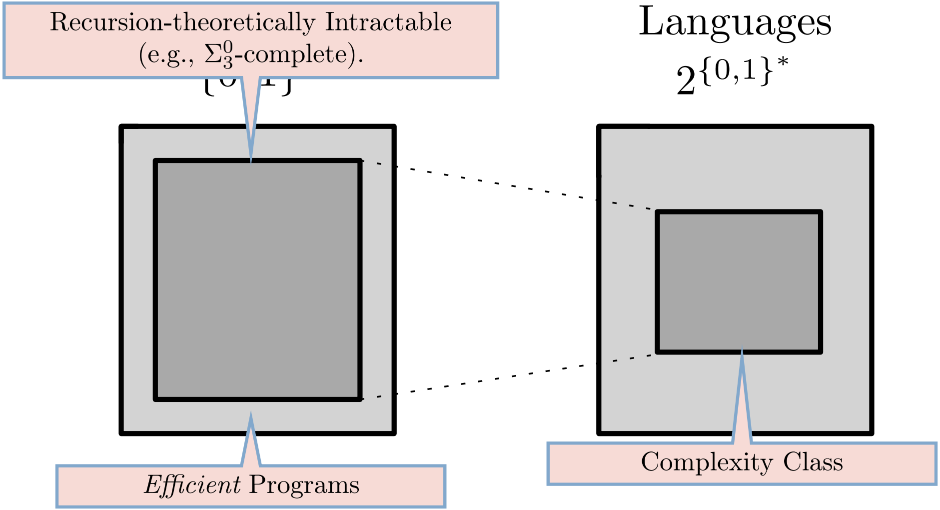
Implicit Computational Complexity



Implicit Computational Complexity



Implicit Computational Complexity



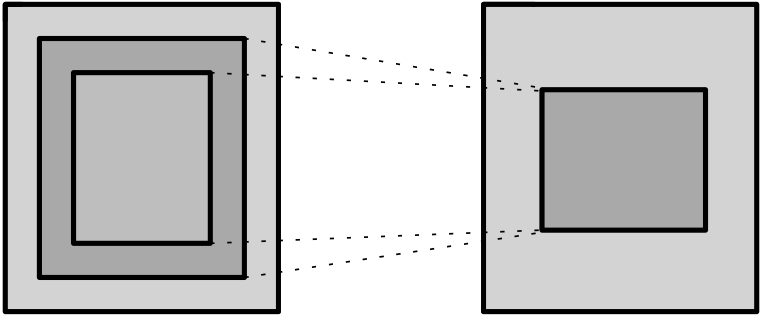
Implicit Computational Complexity

Programs

$\{0,1\}^*$

Languages

$2^{\{0,1\}^*}$



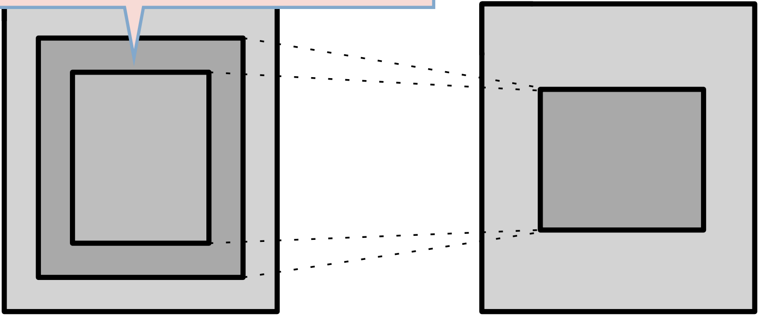
Implicit Computational Complexity

Recursion-theoretically Tractable

Examples: bounded arithmetic, safe recursion, light logics, path orders, type systems, etc.

Languages

$2^{\{0,1\}^*}$



Two Kinds of Zoos

Syntactic Classes

P **PSPACE** **L** ...

coNP **NP** **EXP**

Semantic Classes

BPP ...

ZPP **BQP**

Two Kinds of Zoos

Syntactic Classes

P **PSPACE** **L** **coNP** **NP** **EXP** . . .

- Complete problems exist;
- Hierarchy theorems hold.

Semantic Classes

BPP **ZPP** **BQP** . . .

Two Kinds of Zoos

Syntactic Classes

P **PSPACE** **L** ... **coNP** **NP** **EXP**

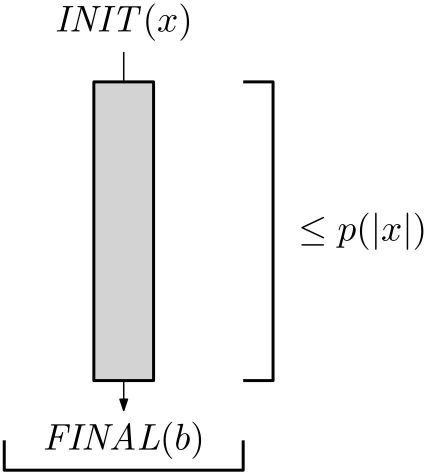
- Complete problems exist;
- Hierarchy theorems hold.

Semantic Classes

BPP **BQP** ... **ZPP**

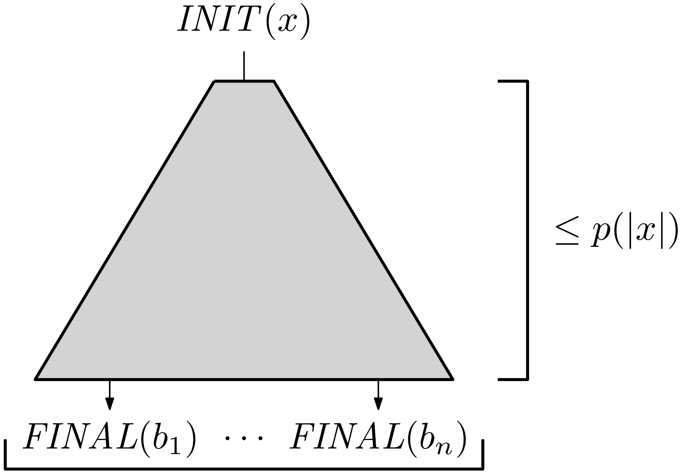
Nothing is known about complete problems nor about hierarchy theorems.

P



$$x \in \mathcal{L} \quad \Leftrightarrow \quad b$$

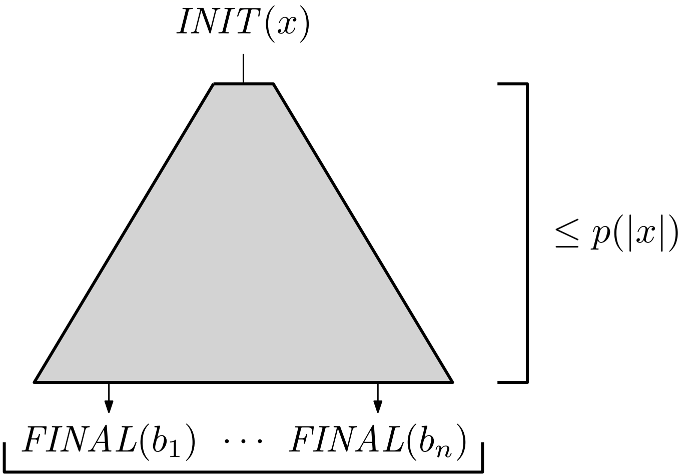
BPP



$$x \in \mathcal{L} \quad \Rightarrow \quad \Pr[b_i] \geq \frac{2}{3}$$

$$x \notin \mathcal{L} \quad \Rightarrow \quad \Pr[\neg b_i] \geq \frac{2}{3}$$

BPP



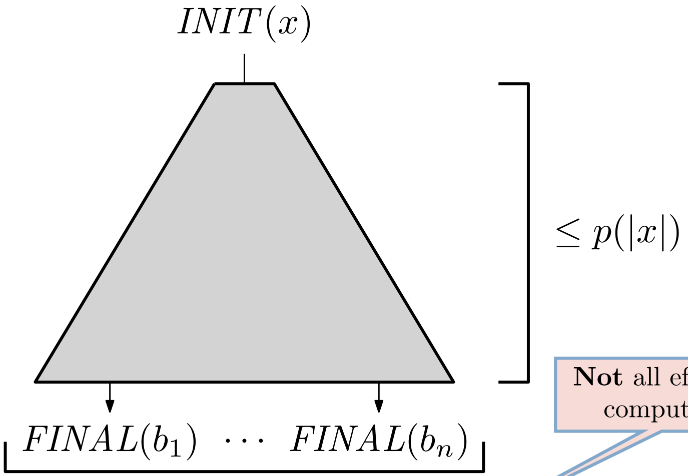
$$x \in \mathcal{L} \quad \Rightarrow \quad \Pr[b_i] \geq \frac{2}{3}$$

$$x \notin \mathcal{L} \quad \Rightarrow \quad \Pr[\neg b_i] \geq \frac{2}{3}$$

Not the same as

$$x \in \mathcal{L} \quad \Leftrightarrow \quad \Pr[b_i] \geq \frac{2}{3}$$

BPP



$$x \in \mathcal{L} \quad \Rightarrow \quad \Pr[b_i] \geq \frac{2}{3}$$
$$x \notin \mathcal{L} \quad \Rightarrow \quad \Pr[\neg b_i] \geq \frac{2}{3}$$

Not all efficient machines compute *a* language!

Not the same as

$$x \in \mathcal{L} \quad \Leftrightarrow \quad \Pr[b_i] \geq \frac{2}{3}$$

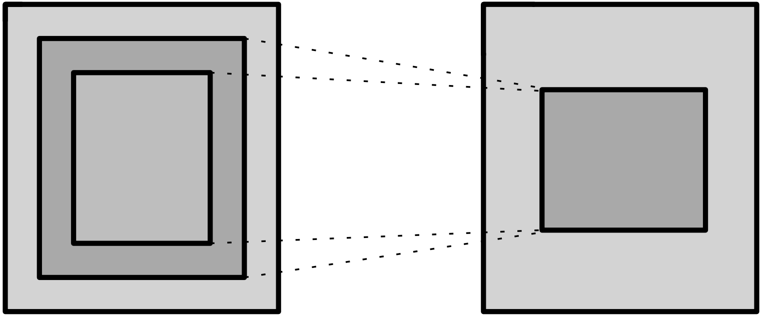
Back to ICC

Programs

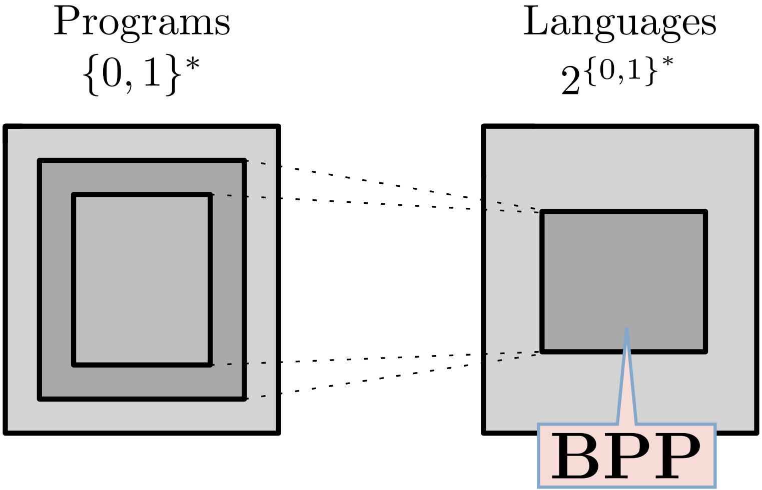
$\{0,1\}^*$

Languages

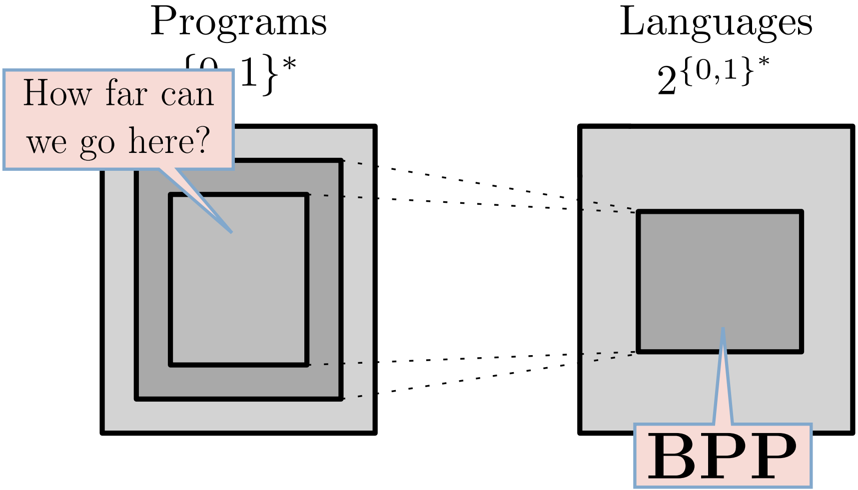
$2^{\{0,1\}^*}$



Back to ICC



Back to ICC



Part II

Bounded Arithmetic

PA as a Way to Represent Functions

$$\mathsf{PA} \vdash \forall x. \exists! y. A(x, y)$$

PA as a Way to Represent Functions

- Peano Axioms.
- Induction holds **in general** and **for every formula**.

$$\mathsf{PA} \vdash \forall x.\exists! y.A(x,y)$$

PA as a Way to Represent Functions

- Peano Axioms.
- Induction holds **in general** and **for every formula**.

$$\mathsf{PA} \vdash \forall x.\exists! y.A(x, y) \quad \Longrightarrow \quad \begin{array}{c} f : \mathbb{S} \to \mathbb{S} \\ \models A(s, f(s)) \text{ for every } s \in \mathbb{S} \end{array}$$

PA as a Way to Represent Functions

- Peano Axioms.
- Induction holds **in general** and **for every formula**.

$$\mathsf{PA} \vdash \forall x. \exists ! y. A(x,y) \qquad \Longrightarrow \qquad \begin{array}{c} f : \mathbb{S} \to \mathbb{S} \\ \models A(s, f(s)) \text{ for every } s \in \mathbb{S} \end{array}$$

$$[\![\mathsf{PA}]\!] := \{ f : \mathbb{S} \to \mathbb{S} \mid f \text{ is provably total in } \mathsf{PA} \}$$

PA as a Way to Represent Functions

- Peano Axioms.
- Induction holds **in general** and **for every formula**.

$$\mathsf{PA} \vdash \forall x.\exists! y.A(x,y) \quad \Longrightarrow \quad \begin{array}{c} f : \mathbb{S} \to \mathbb{S} \\ \models A(s, f(s)) \text{ for every } s \in \mathbb{S} \end{array}$$

$$[\![\mathsf{PA}]\!] := \{f : \mathbb{S} \to \mathbb{S} \mid f \text{ is provably total in } \mathsf{PA}\}$$

Simply **too big a class** for our purposes!

Characterizing **FP**

$$[\![\mathsf{S}^1_2]\!]$$

Characterizing **FP**

$$[\![\mathsf{S}_2^1]\!]$$

- Induction **on notation**.
- Induction formulas are Σ_1^b, namely bounded existential quantifications of sharply bounded formulas.

Characterizing **FP**

- Due to Buss [Buss86].
- Many variations exists.

$$[\![\mathsf{S}^1_2]\!] = \mathbf{FP}$$

Characterizing $\mathbf{FP}$

$$[\![\mathsf{S}^1_2]\!] = \mathbf{FP}$$

$$\begin{tikzcd}
\mathsf{S}^1_2 \arrow[r, bend left] & \mathcal{PR} \arrow[l, bend left] \arrow[r, bend left] & \mathbf{PDTM} \arrow[l, bend left]
\end{tikzcd}$$

Characterizing **FP**

$$\llbracket \mathsf{S}^1_2 \rrbracket = \mathbf{FP}$$

$$\begin{tikzcd}
\mathsf{S}^1_2 \arrow[rr, bend left] & & \mathcal{PR} \arrow[ll, bend left] \arrow[rr, bend left] & & \mathbf{PDTM} \arrow[ll, bend left]
\end{tikzcd}$$

Cobham's **Bounded Recursion on Notation**

Characterizing **FP**

$$[\![\mathsf{S}^1_2]\!] = \mathbf{FP}$$

$$\begin{tikzcd}
\mathsf{S}^1_2 \arrow[r, bend left] & \mathcal{PR} \arrow[l, bend left] \arrow[r, bend left] & \mathbf{PDTM} \arrow[l, bend left]
\end{tikzcd}$$

Cobham's **Bounded Recursion on Notation**

Polytime Deterministic Turing Machines.

Characterizing **FP**

- Arguably the most difficult step.
- Can be done in various ways, e.g. through cut-elimination process, or by realizability.

$$[\![\mathsf{S}^1_2]\!] = \mathbf{FP}$$

$$\begin{tikzcd} \mathsf{S}^1_2 \arrow[r, bend left] & \mathcal{PR} \arrow[l, bend left] \arrow[r, bend left] & \mathbf{PDTM} \arrow[l, bend left] \end{tikzcd}$$

Cobham's **Bounded Recursion on Notation**

Polytime Deterministic Turing Machines.

Part III

Incepting Randomness into BA

$$\begin{array}{c} \mathsf{PA} \\ \Big\downarrow \\ \mathsf{S}^1_2 \end{array}$$

The Main Idea

$$\begin{tikzcd} \mathsf{PA} \arrow[r] \arrow[d] & \mathsf{MQPA} \\ \mathsf{S}^1_2 & \end{tikzcd}$$

The Main Idea

$$\begin{tikzcd} \mathsf{PA} \arrow[r] \arrow[d] & \mathsf{MQPA} \\ S^1_2 & \end{tikzcd}$$

- A conservative extension of PA [CIE2021].
- The **unary predicate** Flip models the access to an oracle providing fair random bits.
- The semantics of a formula is a **measurable set** of truth assignments to $\mathbb{S}$
- **All computable random functions** from $\mathbb{S}$ to distributions over $\mathbb{S}$ can be represented in MQPA.

The Main Idea

$$\begin{tikzcd} \mathsf{PA} \arrow[r] \arrow[d] & \mathsf{MQPA} \arrow[d] \\ \mathsf{S}^1_2 \arrow[r] & \mathsf{RS}^1_2 \end{tikzcd}$$

The Main Idea

$$\begin{tikzcd} \mathsf{PA} \arrow[r] \arrow[d] & \mathsf{MOPA} \arrow[d] \\ \mathsf{S}^1_2 \arrow[r] & \mathsf{RS}^1_2 \end{tikzcd}$$

- The theory stays **essentially the same** as S^1_2, but Flip can now occur in formulas.
- The class $[\![\mathsf{RS}^1_2]\!]$ naturally becomes a class of **random functions**, namely functions from $\mathbb{S}$ to *distributions* over $\mathbb{S}$.

The Result

$$[\![\mathsf{RS}^1_2]\!] \;=\; \{f : \mathbb{S} \to \mathbb{D}(\mathbb{S}) \mid f \text{ can be computed by a } \mathbf{PPTM}\}$$

The Proof

$$
\begin{tikzcd}
\mathsf{RS}^1_2 \arrow[r, bend left] & \mathcal{POR} \arrow[l, bend left] \arrow[r, bend left] & \mathbf{PPTM} \arrow[l, bend left]
\end{tikzcd}
$$

The Proof

$$\begin{tikzcd} \mathsf{RS}^1_2 \arrow[r, bend left] & \mathcal{POR} \arrow[l, bend left] \arrow[r, bend left] & \mathbf{PPTM} \arrow[l, bend left] \end{tikzcd}$$

- Obtained by extending $\mathcal{PR}$ with a basic function accessing the random bit oracle.
- Generates functions from $\mathbb{S} \times 2^{\mathbb{S}}$ to $\mathbb{S}$.

The Proof

- Based on “randomized” realizability.
- Closely follows [CookUrquhart1990].

$$\mathsf{RS}^1_2 \rightleftarrows \mathcal{POR} \rightleftarrows \mathbf{PPTM}$$

- Obtained by extending $\mathcal{PR}$ with a basic function accessing the random bit oracle.
- Generates functions from $\mathbb{S} \times 2^{\mathbb{S}}$ to $\mathbb{S}$.

The Proof

- Based on “randomized” realizability.
- Closely follows [CookUrquhart1990].

- $\mathcal{POR}$ captures functions in $\mathbb{S}^{\mathbb{S}\times 2^{\mathbb{S}}}$;
- **PPTM** rather captures functions in $\mathbb{S}^{\mathbb{S}\times 2^{\mathbb{N}}}$.

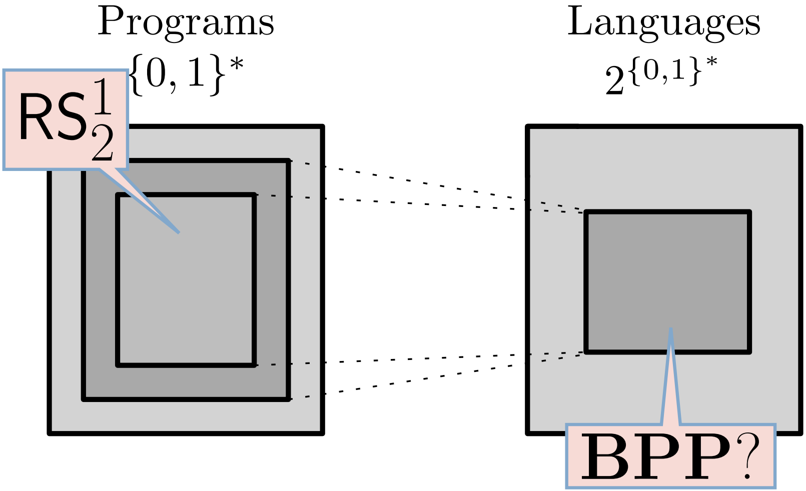
RS^1_2 $\qquad$ $\mathcal{POR}$ $\qquad$ **PPTM**

- Obtained by extending $\mathcal{PR}$ with a basic function accessing the random bit oracle.
- Generates functions from $\mathbb{S} \times 2^{\mathbb{S}}$ to $\mathbb{S}$.

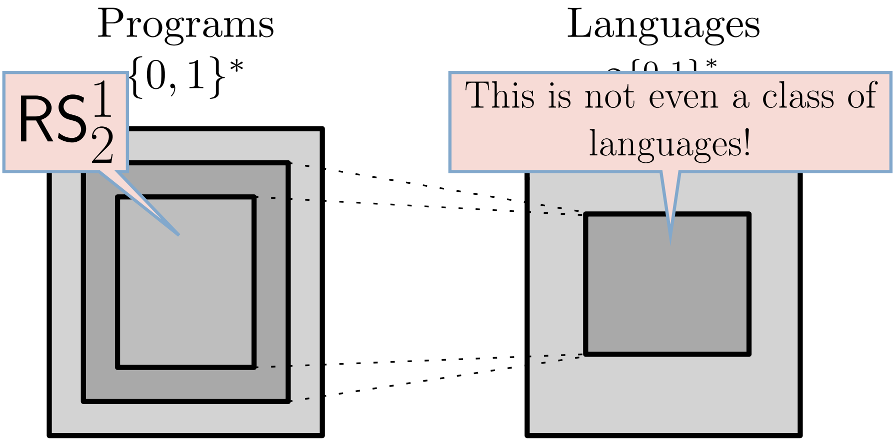
Part IV

Towards **BPP**

Are We There, Yet?



Are We There, Yet? Actually, No!



BPP Through Counting Quantifiers

From...

$$f : \mathbb{S} \to \mathbb{D}(\mathbb{S}) \in [\![\mathsf{RS}^1_2]\!] \quad \Leftrightarrow \quad \begin{array}{c} \mathsf{RS}^1_2 \vdash \forall x.\exists! y.A(x,y) \\ f = \mathit{RandomFunction}(A) \end{array}$$

BPP Through Counting Quantifiers

From. . .

$$f : \mathbb{S} \to \mathbb{D}(\mathbb{S}) \in [\![\mathsf{RS}^1_2]\!] \quad \Leftrightarrow \quad \begin{array}{c} \mathsf{RS}^1_2 \vdash \forall x.\exists! y.A(x,y) \\ f = \mathit{RandomFunction}(A) \end{array}$$

. . . To

$$(L \subseteq \mathbb{S}) \in [\![\mathbf{C}\mathsf{RS}^1_2]\!] \quad \Leftrightarrow \quad \begin{array}{c} \mathsf{RS}^1_2 \vdash \forall x.\exists! y.A(x,y) \\ \models \forall x.\exists y.\mathbf{C}^{\frac{2}{3}} A(x,y) \\ L = \mathit{Language}(A) \end{array}$$

$\mathbf{BPP}$ Through Counting Quantifiers

From…

$$f : \mathbb{S} \to \mathbb{D}(\mathbb{S}) \in [\![\mathsf{RS}^1_2]\!] \quad \Leftrightarrow \quad \begin{array}{c} \mathsf{RS}^1_2 \vdash \forall x.\exists! y.A(x,y) \\ f = \mathit{RandomFunction}(A) \end{array}$$

Counting Quantifier

$$[\![\mathbf{C}^{\frac{t}{s}} B]\!] = \begin{cases} 2^{\mathbb{S}} & \text{if } \mu[\![B]\!] \geq \frac{[\![t]\!]}{[\![s]\!]} \\ \emptyset & \text{otherwise} \end{cases}$$

$$(L \subseteq \mathbb{S}) \in [\![\mathsf{CRS}^1_2]\!] \quad \Leftrightarrow \quad \begin{array}{c} \mathsf{RS}_2 \vdash \forall x.\exists! y.A(x,y) \\ \models \forall x.\exists y.\mathbf{C}^{\frac{2}{3}} A(x,y) \\ L = \mathit{Language}(A) \end{array}$$

$\mathbf{BPP}$ Through Counting Quantifiers

From...

$$f : \mathbb{S} \to \mathbb{D}(\mathbb{S}) \in [\![\mathsf{RS}^1_2]\!] \quad \Leftrightarrow \quad \begin{array}{c} \mathsf{RS}^1_2 \vdash \forall x.\exists! y.A(x,y) \\ f = \mathit{RandomFunction}(A) \end{array}$$

... To

$$(L \subseteq \mathbb{S}) \in [\![\mathbf{C}\mathsf{RS}^1_2]\!] \quad \Leftrightarrow \quad \begin{array}{c} \mathsf{RS}^1_2 \vdash \forall x.\exists! y.A(x,y) \\ \models \forall x.\exists y.\mathbf{C}^{\frac{2}{3}} A(x,y) \\ L = \mathit{Language}(A) \end{array}$$

Theorem

$$[\![\mathbf{C}\mathsf{RS}^1_2]\!] = \mathbf{BPP}$$

Getting Rid of Counting Quantification

From...

$$(L \subseteq \mathbb{S}) \in [\![\mathbf{C}\mathsf{RS}^1_2]\!] \quad \Leftrightarrow \quad \begin{array}{c} \mathsf{RS}^1_2 \vdash \forall x.\exists! y.A(x,y) \\ \models \forall x.\exists y.\mathbf{C}^{\frac{2}{3}} A(x,y) \\ L = Language(A) \end{array}$$

Getting Rid of Counting Quantification

From...

$$(L \subseteq \mathbb{S}) \in [\![\mathsf{CRS}^1_2]\!] \quad \Leftrightarrow \quad \begin{array}{c} \mathsf{RS}^1_2 \vdash \forall x.\exists! y.A(x,y) \\ \models \forall x.\exists y.\mathbf{C}^{\frac{2}{3}} A(x,y) \\ L = \mathit{Language}(A) \end{array}$$

... To

$$(L \subseteq \mathbb{S}) \in [\![\mathsf{T} \oplus \mathsf{RS}^1_2]\!] \quad \Leftrightarrow \quad \begin{array}{c} \mathsf{RS}^1_2 \vdash \forall x.\exists! y.A(x,y) \\ \mathsf{T} \vdash \forall x.\exists y.\mathsf{TwoThirds}[A](x,y) \\ L = \mathit{Language}(A) \end{array}$$

Getting Rid of Counting Quantification

From. . .

$$\mathsf{RS}^1_2 \vdash \forall x.\exists! y.A(x,y)$$

$$(L \subseteq \mathbb{S}) \in [\![\mathsf{CRS}^1_2]\!] \quad \Leftrightarrow \quad \vdash \forall x.\exists y.\mathbf{C}^{\frac{2}{3}} A(x,y)$$

- We can **internalize** Error Bounds into plain arithmetic, making `Flip` to disappear.
- This goes via **threshold quantifiers**.

$$\mathsf{RS}^1_2 \vdash \forall x.\exists! y.A(x,y)$$

$$(L \subseteq \mathbb{S}) \in [\![\mathsf{T} \oplus \mathsf{RS}^1_2]\!] \quad \Leftrightarrow \quad \mathsf{T} \vdash \forall x.\exists y.\mathsf{TwoThirds}[A](x,y)$$

$$L = Language(A)$$

Getting Rid of Counting Quantification

From...

$$(L \subseteq \mathbb{S}) \in [\![\mathsf{CRS}^1_2]\!] \quad \Leftrightarrow \quad \begin{array}{c} \mathsf{RS}^1_2 \vdash \forall x.\exists! y.A(x,y) \\ \models \forall x.\exists y.\mathbf{C}^{\frac{2}{3}} A(x,y) \\ L = \mathit{Language}(A) \end{array}$$

... To

$$(L \subseteq \mathbb{S}) \in [\![\mathsf{T} \oplus \mathsf{RS}^1_2]\!] \quad \Leftrightarrow \quad \begin{array}{c} \mathsf{RS}^1_2 \vdash \forall x.\exists! y.A(x,y) \\ \mathsf{T} \vdash \forall x.\exists y.\mathsf{TwoThirds}[A](x,y) \\ L = \mathit{Language}(A) \end{array}$$

Theorem

$$\forall \mathsf{T}.\ [\![\mathsf{T} \oplus \mathsf{RS}^1_2]\!] \subseteq \mathbf{BPP}$$

Getting Rid of Counting Quantification

From...

$$(L \subseteq \mathbb{S}) \in [\![\mathsf{CRS}^1_2]\!] \quad \Leftrightarrow \quad \begin{array}{c} \mathsf{RS}^1_2 \vdash \forall x.\exists! y.A(x,y) \\ \models \forall x.\exists y.\mathbf{C}^{\frac{2}{3}} A(x,y) \\ L = \mathit{Language}(A) \end{array}$$

... To

$$(L \subseteq \mathbb{S}) \in [\![\mathsf{T} \oplus \mathsf{RS}^1_2]\!] \quad \Leftrightarrow \quad \begin{array}{c} \mathsf{RS}^1_2 \vdash \forall x.\exists! y.A(x,y) \\ \mathsf{T} \vdash \forall x.\exists y.\mathsf{TwoThirds}[A](x,y) \\ L = \mathit{Language}(A) \end{array}$$

Theorem

$$\forall \mathsf{T}.\ [\![\mathsf{T} \oplus \mathsf{RS}^1_2]\!] \subseteq \mathbf{BPP}$$

Theorem

$$\mathtt{PIT} \in [\![\mathsf{PA} \oplus \mathsf{RS}^1_2]\!]$$

Getting Rid of Counting Quantification

From. . .

$$(L \subseteq \mathbb{S}) \in [\![\mathbf{CRS}^1_2]\!] \quad \Leftrightarrow \quad \begin{array}{c} \mathsf{RS}^1_2 \vdash \forall x.\exists! y.A(x,y) \\ \models \forall x.\exists y.\mathbf{C}^{\frac{2}{3}} A(x,y) \\ L = Language(A) \end{array}$$

To. . .

- **Polynomial Identity Testing**.
- In **BPP**, but currently not known to be in **P**.

$$(L \subseteq \mathbb{S} \quad \begin{array}{c} \mathsf{RS}^1_2 \vdash \forall x.\exists! y.A(x,y) \\ \forall x.\exists y.\mathsf{TwoThirds}[A](x,y) \\ L = Language(A) \end{array}$$

Theorem

$$\forall \mathsf{T}.\ [\![\mathsf{T} \oplus \mathsf{RS}^1_2]\!] \subseteq \mathbf{BPP}$$

Theorem

$$\mathtt{PIT} \in [\![\mathsf{PA} \oplus \mathsf{RS}^1_2]\!]$$

Wrapping Up

- ICC and bounded arithmetic can be seen as ways to *enumerate* complexity classes by simple enough languages, thus revealing their structure.
- Semantic classes like **BPP** are not known to be enumerable, due to the *error bound* intrinsic in their definitions.
- We can however enumerate *subclasses* of **BPP** by *internalizing* the error bound check.
- What would be the consequences of $[\![\mathsf{PA} \oplus \mathsf{RS}^1_2]\!] = \mathbf{BPP}$?

Wrapping Up

- ICC and bounded arithmetic can be seen as ways to *enumerate* complexity classes by simple enough languages, thus revealing their structure.
- Semantic classes like **BPP** are not known to be enumerable, due to the *error bound* intrinsic in their definitions.
- We can however enumerate *subclasses* of **BPP** by *internalizing* the error bound check.
- What would be the consequences of $[\![\mathsf{PA} \oplus \mathsf{RS}^1_2]\!] = \mathbf{BPP}$?

Thank you! Questions?