Bounded Arithmetic I: Provably Total Functions

Sam Buss

Caleidoscope Research School
Institute Henri Poincaré, Paris
June 17-18, 2019

A quote

Stephen Cook, 1975, Feasibly constructive proofs and the propositional calculus

> *A constructive proof of, say, a statement $\forall x A$ must provide an effective means of finding a proof of A for each value of x, but nothing is said about how long this proof is as a function of x. If the function is exponential or super exponential, then for short values of x the length of the proof of the instance of A may exceed the number of electrons in the universe.*

In the paper introducing PV and the Cook translation

Bounded arithmetic gives a rich perspective on and a different approach to fundamental questions in computational complexity from the point of view of mathematical logic.

It joins the study of

feasible computability and complexity

with questions about

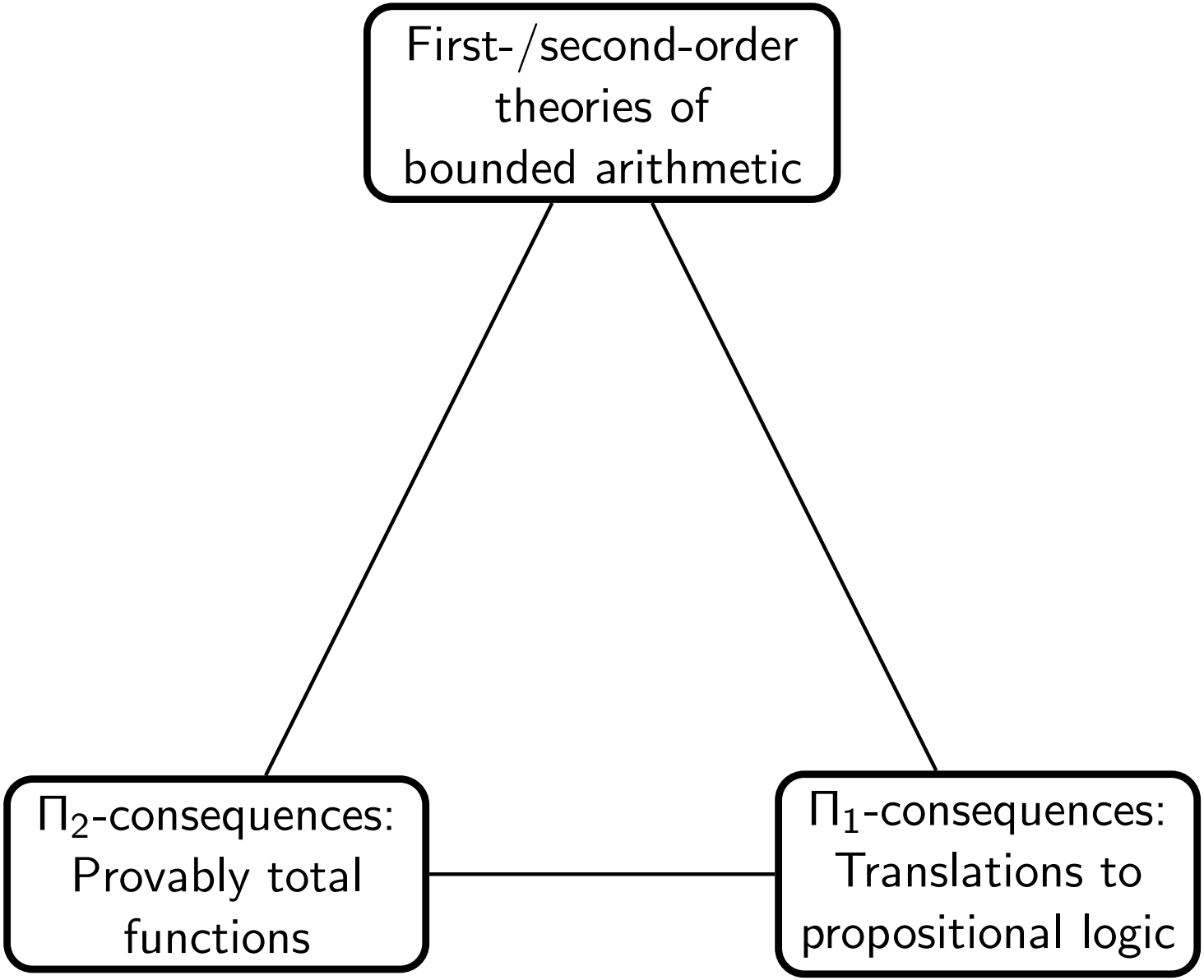
provability and axiomatizability.

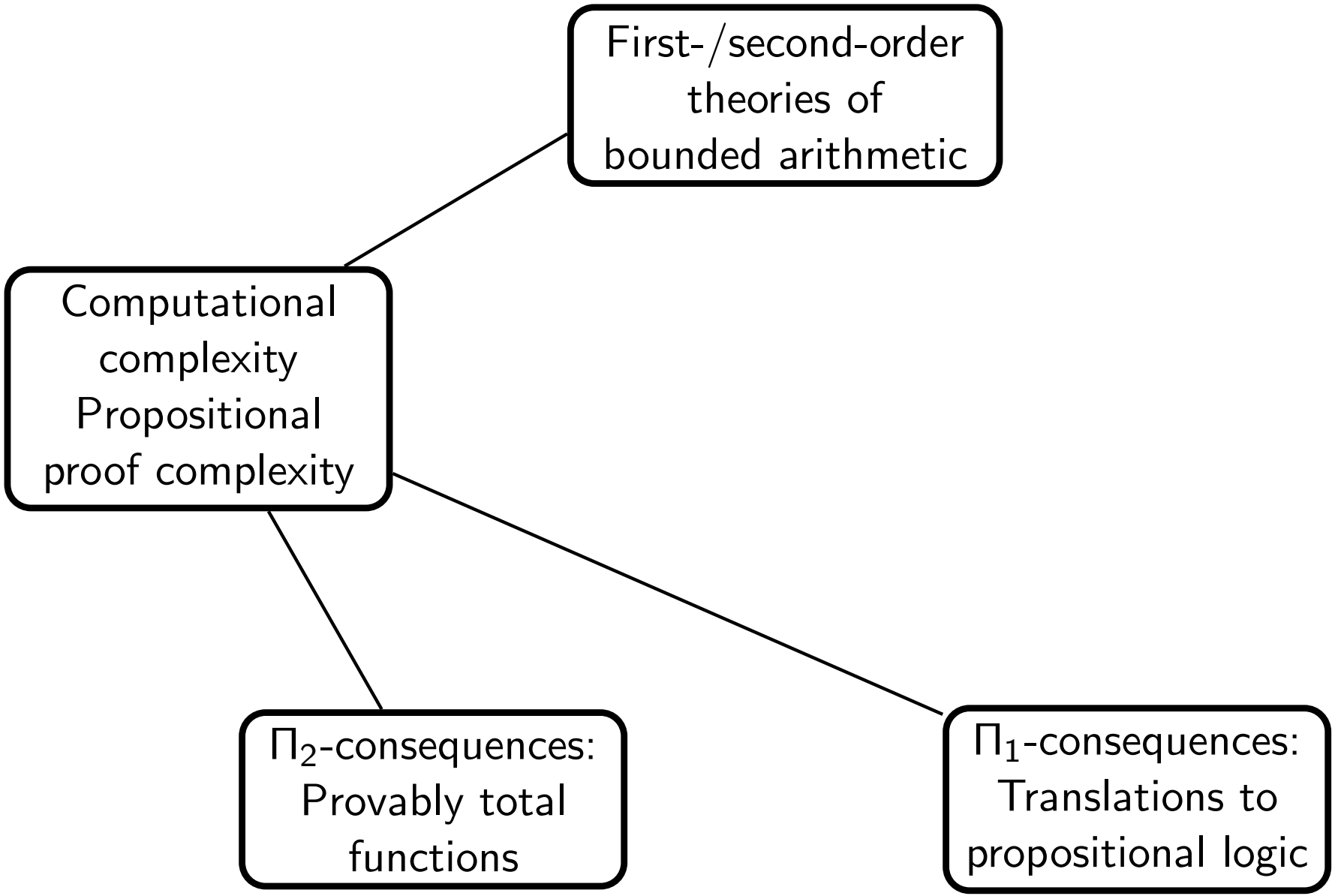
with close connections to

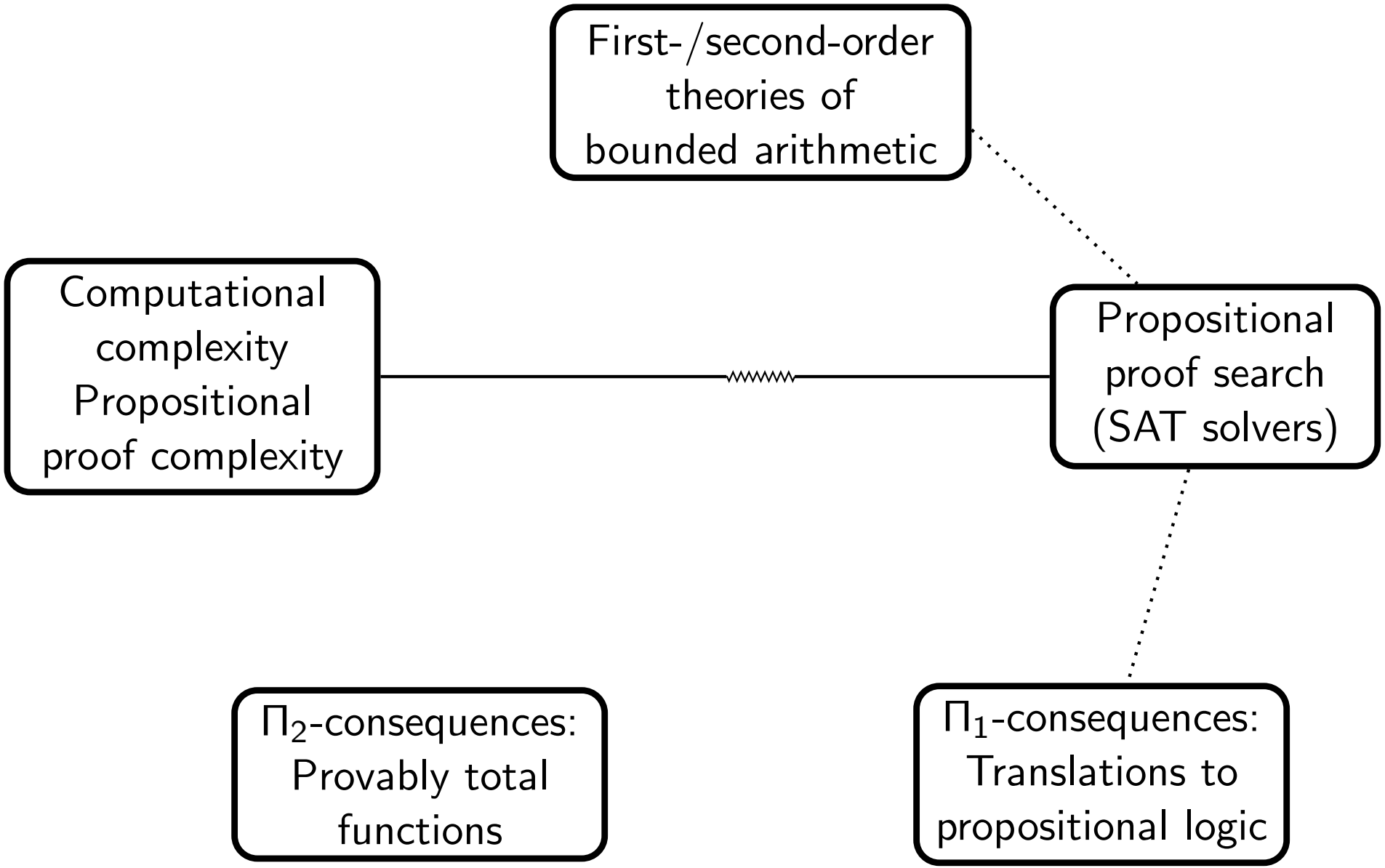
propositional proof complexity.

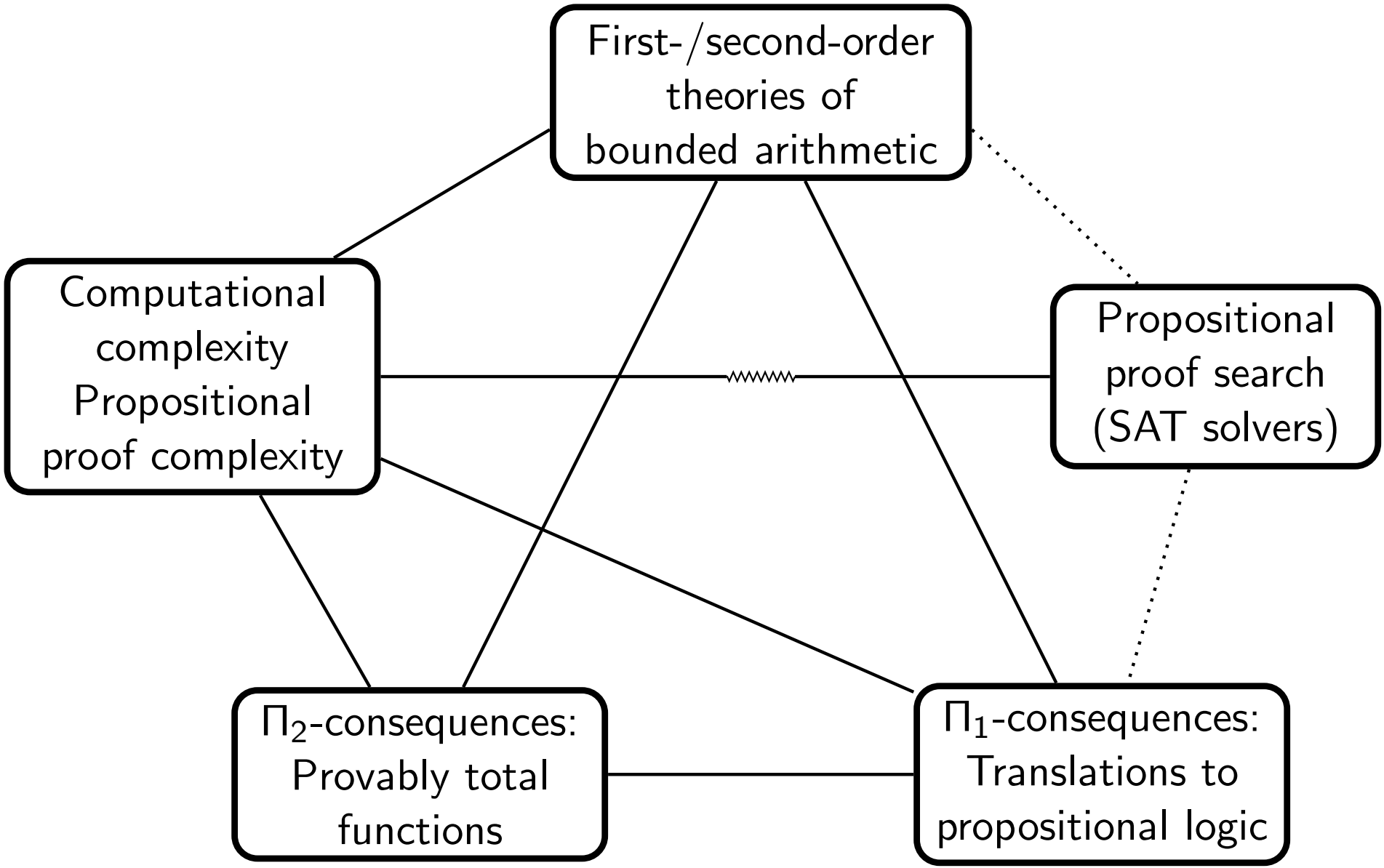
Bounded Arithmetic Theories S^i_2 and T^i_2 and more.

- Weak feasible fragments of Peano arithmetic, and Primitive Recursive Arithmetic. Formulated with restricted induction axioms.
- Have close connections to "feasible" complexity classes (e.g., P, polynomial time; (non-)deterministic logspace, L, NL; alternating log time, $\mathrm{ALogTime}$), and near-feasible complexity classes (e.g., Polynomial Local Search, PLS; the polynomial time hierarchy or PSPACE).
- Have close connections to propositional proof systems.
- Have close connections to open problems in computational complexity. (E.g., P versus NP, the polynomial time hierarchy, the existence of pseudorandom number generators, and the hardness of TFNP, Total NP search problems).


First-/second-order theories of bounded arithmetic
Π_2-consequences: Provably total functions
Π_1-consequences: Translations to propositional logic


First-/second-order theories of bounded arithmetic
Computational complexity Propositional proof complexity
Π_2-consequences: Provably total functions
Π_1-consequences: Translations to propositional logic


First-/second-order theories of bounded arithmetic
Computational complexity Propositional proof complexity
Propositional proof search (SAT solvers)
Π_2-consequences: Provably total functions
Π_1-consequences: Translations to propositional logic


First-/second-order theories of bounded arithmetic
Computational complexity Propositional proof complexity
Propositional proof search (SAT solvers)
Π_2-consequences: Provably total functions
Π_1-consequences: Translations to propositional logic

First-order bounded arithmetic, bounded quantifiers

Language of bounded arithmetic includes:

$$0, \quad S, \quad +, \quad \cdot, \quad \leq, \quad |x|, \quad \lfloor \tfrac{1}{2}x \rfloor, \quad x\#y, \quad \mathrm{MSP}(x,i).$$

where

$|x| :=$ length of binary representation of x.
$x\#y := 2^{|x|\cdot|y|}$; so $|x\#y| = |x| \cdot |y| + 1$.
$\mathrm{MSP}(x,i) := \lfloor x/2^i \rfloor$. ("most significant part")

Symbols for Peano Arithmetic plus:

- $x\#y$ gives polynomial growth rate functions.
- MSP gives simple sequence coding using binary representation.
- $|x|$ and $\lfloor \frac{1}{2}x \rfloor$ - facilitate "feasible" forms of induction.

Definition

- **Bounded Quantifier:** of the form $(\forall x \leq t)$ or $(\exists x \leq t)$.
- **Sharply Bounded Quantifier:** of the form $(\forall x \leq |t|)$ or $(\exists x \leq |t|)$.

Definition

A formula is **bounded** or **sharply bounded** provided all its quantifiers are bounded or sharply bounded (resp.).

Definition (Quantifier alternation classes)

$\Delta_0^b = \Sigma_0^b = \Pi_0^b$: Sharply bounded formulas

Σ_{i+1}^b: Closure of Π_i^b under existential bounded quantification and arbitrary sharply bounded quantification, modulo prenex operations.

Π_{i+1}^b is defined dually.

Connections with polynomial time and the polynomial time hierarchy:

- All terms $t(x)$ have polynomial growth rate: $|t(x)| = |x|^{O(1)}$.
- Sharply bounded formulas ($\Delta_0^{\mathrm{b}} = \Sigma_0^{\mathrm{b}} = \Pi_0^{\mathrm{b}}$) are polynomial time predicates.
- Σ_1^{b}-formulas define exactly NP properties.
- Π_1^{b}-formulas define exactly coNP properties.
- Σ_i^{b}- and Π_i^{b}-formulas define exactly the predicates in the classes Σ_i^p and Π_i^p at the i-th level of the polynomial time hierarchy.

Why include # (smash)?

- Gives terms of polynomial growth rate; hence connections with the polynomial time hierarchy.
- Gives the growth rate needed for convenient arithmetization of metamathematics. (E.g., the operation of substitution requires polynomial growth rate.)
- Gives a **quantifier exchange property** (together with MSP)

$$(\forall x{<}|t|)(\exists y{<}s)A(x,y) \rightarrow (\exists w{<}t\#s)(\forall x{<}|t|)A(x,(w)_x)$$

for suitable Gödel decoding function $(\cdot)_x$

Axioms for bounded arithmetics:

BASIC: A set of open (quantifier-free) statements defining simple properties of the function symbols. For example,

$x + (y + z) = (x + y) + z$ $\qquad\qquad$ $\mathrm{MSP}(x, S(i)) = \lfloor \frac{1}{2}\mathrm{MSP}(x, i) \rfloor$.

Induction axioms: Letting A range over Φ-formulas,

Φ-IND: $\quad A(0) \wedge (\forall x)(A(x) \to A(x{+}1)) \to (\forall x)A(x)$.

Φ-PIND: $\quad A(0) \wedge (\forall x)(A(\lfloor \frac{1}{2}x \rfloor) \to A(x)) \to (\forall x)A(x)$.

Φ-LIND: $\quad A(0) \wedge (\forall x)(A(x) \to A(x{+}1)) \to (\forall x)A(|x|)$.

Φ-PIND and Φ-LIND are "polynomially feasible" versions of induction.

The theories S_2^i and T_2^i

Definition (Fragments of bounded arithmetic, B'85)

S_2^i: $\text{BASIC} + \Sigma_i^b$-PIND.
T_2^i: $\text{BASIC} + \Sigma_i^b$-IND.

$S_2 = \cup_i S_2^i$ and $T_2 = \cup_i T_2^i$.

Note: T_2 is essentially $I\Delta_0 + \Omega_1$. [Parikh'71, Wilkie-Paris'87]

Theorem (B'85, B'90)

(a) $S_2^1 \subseteq T_2^1 \preccurlyeq_{\forall\Sigma_2^b} S_2^2 \subseteq T_2^2 \preccurlyeq_{\forall\Sigma_3^b} S_2^3 \subseteq \cdots$
(b) *Thus,* $S_2 = T_2$.

Proof that $\mathrm{T}_2^i \supset \mathrm{S}_2^i$:

Lemma

Σ_i^{b}-PIND *follows from* Σ_i^{b}-LIND *(over* BASIC, $i \geq 1$*).*

Proof: (Sketch) To prove PIND for $A(x)$, (with c a free variable)

$$A(0) \wedge (\forall x)(A(\lfloor \tfrac{1}{2}x \rfloor) \rightarrow A(x)) \rightarrow A(c)$$

use LIND on $B(i) := A(t(i))$ for $t(i) := \mathrm{MSP}(c, |c|-i)$.
For this, note $B(0)$ and $B(|c|)$ are equivalent to $A(0)$ and $A(c)$.
Also, $t(i) = \lfloor \frac{1}{2}t(i+1) \rfloor$, so $(\forall i)(B(i) \rightarrow B(i+1))$ follows from $(\forall x)(A(\lfloor \frac{1}{2}x \rfloor) \rightarrow A(x))$. □

Corollary

$\mathrm{S}_2^i \subset \mathrm{T}_2^i$, *for* $i \geq 1$.

Provably total functions and Σ^b_i-definable functions

Definition

A function $f : \mathbb{N} \to \mathbb{N}$ is **provably total** in a theory R provided there is a formula $A_f(x, y)$ satisfying

- $A_f(x, y)$ defines the graph of $f(x) = y$
- R proves $(\forall x)(\exists! y)A_f(x, y)$
- A_f is polynomial time computable.

Definition

f is Σ^b_1-**definable by** R, provided there is a Σ^b_1-formula $A(x, y)$ such that

- $R \vdash (\forall x)(\exists y \leq t)A(x, y)$ for some term t.
- $R \vdash (\forall x, y, y')(A(x, y) \wedge A(x, y') \to y = y')$.
- $A(x, y)$ defines the graph of f.

"Σ^b_i-definable" is defined similarly, but allowing $A \in \Sigma^b_i$.

Σ_1^b-definability is more important than Σ_1^b-definability.

Theorem

Any Σ_1^b-definable function in S_2^i or T_2^i can be introduced conservatively into the language of the theory with its defining axiom, and be used freely in induction formulas.

Theorem (B'85)

S_2^1 can Σ_1^b-define every polynomial time function.

(The converse holds too, as is discussed later.)

Hence, we can w.l.o.g. assume that all polynomial time functions are present in the language of any bounded arithmetic theory containing S_2^1..

Similar definitions and results hold for predicates.

Main Theorem for S^1_2

The converses of the last theorems also hold: S^1_2 can Σ^{b}_1-define *exactly* the polynomial time functions.

Theorem (Main Theorem for S^1_2, B'85)

Suppose f is Σ^{b}_1-defined by S^1_2. Then f is computable in polynomial time.
In fact, S^1_2 can prove f is computed by a polynomial time Turing machine.

The corresponding theorem for predicates:

Theorem

The Δ^{b}_1-definable predicates of S^1_2 are precisely the predicates that are S^1_2-provably in P.

These show S^1_2 has proof-theoretic strength corresponding to polynomial time computation.

- The proof of the "Main Theorem for S_2^1" uses a "witnessing" argument.
- Applying cut elimination, there is a sequent calculus proof P of the sequent

$$\longrightarrow (\exists y \leq t(c)) A(c, y).$$

  in which every formula is Σ_1^{b}.
- The sequent calculus proof P can be read as a **computer program** for computing a y as a function c, together with a **proof of correctness** of the program.
- The program has polynomial runtime.
- The **PIND inferences** in the proof P correspond to **for-loops**.

The next slides spell out some of the details....

Intuition for the proof of the "Main Theorem for S_2^1"

- Using free-cut elimination, we have a proof in which every line has the form $\Gamma \longrightarrow \Delta$:

$$A_1, A_2, \ldots, A_k \longrightarrow B_1, B_2, \ldots, B_\ell$$

  in which every formula is Σ_1^{b}.
- Then prove that, for each such sequent, there is a polynomial time function

$$f(u_1, \ldots, u_k) = \langle j, v \rangle,$$

  which given u_i's witnessing the outer existential quantifiers of the A_i', produces a pair (j, v) so that v witnesses the outermost quantifier of B_j.

It considerably simplifies the proof to work with special subclasses of *prenex* formulas:

- **Strict Σ^{b}_i formulas ($\mathrm{s}\Sigma^{\mathrm{b}}_i$)**: Of the form

$$(\exists x_1 \leq t_1)(\forall x_2 \leq t_2) \cdots (Q x_i \leq t_i) B(\vec{x}),$$

  where B is sharply bounded. (And subformulas of these.)
- **Sharply strict Σ^{b}_i formulas ($\mathrm{ss}\Sigma^{\mathrm{b}}_i$)**: Of the form

$$(\exists x_1 \leq t_1)(\forall x_2 \leq t_2) \cdots (Q x_i \leq t_i)(\overline{Q} x_{i+1} \leq |t_{i+1}|) B(\vec{x}),$$

  where B is quantifier free. (And subformulas of these.)

Proposition:

- Every Σ^{b}_i formula is equivalent to an $\mathrm{ss}\Sigma^{\mathrm{b}}_i$ formula (provably in S^1_2).
- S^i_2 may be equivalently formalized with $\mathrm{ss}\Sigma^{\mathrm{b}}_i$-PIND ($i \geq 1$).

To prove the witnessing theorem, by free-cut elimination, it suffices to consider sequent calculus proofs in which every formula is an $\mathrm{ss}\Sigma^{\mathrm{b}}_1$-formula.

Definition

Let $A(\vec{c})$ be $\mathrm{ss}\Sigma^{\mathrm{b}}_1$. The predicate $Wit_A(\vec{c}, u)$ is defined so that

- If A is $(\exists x{\leq}t)B(\vec{c}, x)$, $B \in \Delta^{\mathrm{b}}_0$, then $Wit_A(\vec{c}, u)$ is the formula $u \leq t \wedge B(\vec{c}, u)$.
- If A is in Δ^{b}_0, then $Wit_A(\vec{c}, u)$ is just $A(\vec{c})$.

We have immediately

Fact: $A(\vec{c}) \leftrightarrow (\exists u) Wit_A(\vec{c}, u)$.

Fact: Wit_A is a Δ^{b}_0-formula (and thus polynomial time).

Theorem (Witnessing Lemma for S^1_2)

Suppose S^1_2 *proves sequent*

$$A_1, A_2, \ldots, A_k \longrightarrow B_1, B_2, \ldots, B_\ell$$

of $\mathrm{ss}\Sigma^{\mathrm{b}}_1$ *formulas with free variables* $\vec{c}$*, then there is a* Σ^{b}_1*-definable function* $f(\vec{c}, \vec{u})$ *which is provably computable in polynomial time such that* S^1_2 *proves*

$$\textit{If } \bigwedge\nolimits_i \mathit{Wit}_{A_i}(\vec{c}, u_i) \textit{ then } f(\vec{c}, \vec{u}) = \langle j, v\rangle \textit{ where } \mathit{Wit}_{B_j}(\vec{c}, v).$$

$$\textit{I.e.,} \quad \bigwedge\nolimits_i \mathit{Wit}_{A_i}(\vec{c}, u_i) \longrightarrow \bigvee\nolimits_j \left[(f(\vec{c}, \vec{u}))_1 = j \wedge \mathit{Wit}_j(\vec{c}, (f(\vec{c}, \vec{u}))_2)\right].$$

(Subscripts "1" and "2" represent the Gödel beta function.)
The Witnessing Lemma will be proved by induction on the number of lines in a free-cut free S^1_2-proof P of $\Gamma \longrightarrow \Delta$.

The Main Theorem for S^1_2 is an immediate corollary. $(\square)$

Case (1): Last inference is $\exists \leq$:right.

$$\frac{\Gamma \longrightarrow \Delta, A(\vec{c}, s)}{s \leq t, \Gamma \longrightarrow \Delta, (\exists x \leq t) A(\vec{c}, x)}$$

The formula A is $\mathrm{ss}\Delta_0^{\mathrm{b}}$. The induction hypothesis gives a function f, which accepts witnesses for Γ and produces a witness either making a formula in Δ true or indicating $A(\vec{c}, s)$ true. Modify f, so that in the latter case, it returns $\langle \ell, s \rangle$.

$$g(\vec{c}, u) \;=\; \begin{cases} f(\vec{c}, cdr(u)) & \text{if } (f(\vec{c}, cdr(u)))_1 < \ell \\ \langle \ell, s(\vec{c}) \rangle & \text{if } (f(\vec{c}, cdr(u)))_1 = \ell. \end{cases}$$

(The "*cdr*" operation strips the first entry from a sequence.)

Case (2): Last inference is $\exists \leq$:left.

$$\frac{b \leq t, A(\vec{c}, b), \Gamma \longrightarrow \Delta}{(\exists x \leq t) A(\vec{c}, x), \Gamma \longrightarrow \Delta}$$

where A is $\mathrm{ss}\Delta_0^{\mathrm{b}}$. Let f be given by the induction hypothesis. Define g by

$$g(\vec{c}, u) \;=\; f(\vec{c}, (u)_1, \langle 0 \rangle * u)$$

(The "$*$" operation is sequence concatenation: The "0" serves to witness $b \leq t$.)

Case (3): Last inference is a Cut.

$$\frac{\Gamma \longrightarrow \Delta, C \qquad C, \Gamma \longrightarrow \Delta}{\Gamma \longrightarrow \Delta}$$

where C is an $\mathrm{ss}\Sigma^{\mathrm{b}}_1$-formula. The induction hypothesis gives two functions f and g for the upper sequents. We compose them to form

$$h(\vec{c}, \vec{u}) \;=\; \begin{cases} f(\vec{c}, \vec{u}) & \text{if } (f(\vec{c}, \vec{u})_1 < \ell) \\ g(\vec{c}, \langle (f(\vec{c}, \vec{u}))_2 \rangle * \vec{u}) & \text{otherwise} \end{cases}$$

In other words, if the first function, f, gives a witness for C instead of Δ, then this is used to apply the second function, g.

Case (4): Last inference is PIND.

$$\frac{A(\lfloor \frac{1}{2} b \rfloor), \Gamma \longrightarrow \Delta, A(b)}{A(0), \Gamma \longrightarrow \Delta, A(t)}$$

where $A \in \mathrm{ss}\Sigma^{\mathrm{b}}_1$. This is handled by iterating the construction for Cut.

$$h(\vec{c}, b, u) = \begin{cases} h(\vec{c}, \lfloor \frac{1}{2} b \rfloor, u) & \text{if } (h(\vec{c}, \lfloor \frac{1}{2} b \rfloor, u))_1 < \ell \\ f(\vec{c}, b, \langle (h(\vec{c}, \lfloor \frac{1}{2} b \rfloor, u))_2 \rangle * cdr(u)) & \text{otherwise} \end{cases}$$

and $h(\vec{c}, 0, u) = \langle \ell, (u)_1 \rangle$. h can be defined by *limited iteration on notation* and is polynomial time computable relative to f.
Then set $g(\vec{c}, u) = h(\vec{c}, t(\vec{c}), u)$.

Other cases omitted; the only nontrivial case remaining is $\forall$:right.

Q.E.D. Witnessing Lemma and Theorem.

Generalizations to $i > 1$.

Theorem (B'85)

Let $i \geq 1$. S^i_2 can Σ^{b}_i-define every function which is polynomial time computable with an oracle from Σ^p_{i-1}.

Recall that for $i = 1$ this gave just the polynomial time functions.

Conversely:

Theorem (Main Theorem for S^i_2, B'85)

Let $i \geq 1$. Suppose f is Σ^{b}_i-defined by S^i_2. Then f is computable in $P^{\Sigma^p_{i-1}}$, that is, in polynomial time with an oracle for Σ^p_{i-1}.

Recall:

$$\mathrm{S}^1_2 \subseteq \mathrm{T}^1_2 \preccurlyeq_{\forall\Sigma^{\mathrm{b}}_2} \mathrm{S}^2_2 \subseteq \mathrm{T}^2_2 \preccurlyeq_{\forall\Sigma^{\mathrm{b}}_3} \mathrm{S}^3_2 \subseteq \cdots$$

Theorem (B'90)

Let $i \geq 1$.

1. S^{i+1}_2 *is* $\forall\exists^{\mathrm{b}}_{i+1}$*-conservative over* T^i_2.
2. *In particular,* T^i_2 *can* $\Sigma^{\mathrm{b}}_{i+1}$ *define precisely the functions in* $P^{\Sigma^{\mathrm{b}}_i}$.

Proof idea:

- First show that T^i_2 can $\Sigma^{\mathrm{b}}_{i+1}$ define the functions in $P^{\Sigma^{\mathrm{b}}_i}$.
- Second, show that T^i_2 can prove (each instance of) the Witnessing Lemma for S^{i+1}_2.

Theorem (Krajíček-Pudlák-Takeuti'91, B'95, Zambella'96)

If $\mathrm{T}^i_2 = \mathrm{S}^{i+1}_2$*, then the polynomial time hierarchy collapses (provably) — to* $\Sigma^p_{i+1}/poly$ *and to* $B(\Sigma^{\mathrm{b}}_{i+2})$.

Pause

Next: The Σ^{b}_1-definable functions of T^1_2 are the PLS functions.

Polynomial Local Search (PLS)

Inspired by Dantzig's algorithm and other local search algorithms:

Definition (JPY'88.)

A PLS problem consists of polynomial time functions: $N(x,s)$, $i(x)$, and $c(x,s)$, polynomial time predicate $F(x,s)$, and polynomial bound $b(x) \leq 2^{|x|^{O(1)}}$ such that

0. $\forall x(F(x,s) \rightarrow s \leq b(x))$.
1. $\forall x(F(x,i(x)))$.
2. $\forall x(N(x,s) = s \vee c(x,N(x,s)) < c(x,s))$.
3. $\forall x(F(x,s) \rightarrow F(x,N(x,s)))$.

The input is x.
A **solution** is a point s such that $F(x,s)$ and $N(x,s) = s$.

i - initial point. c - cost function. N - neighbor function.
A solution is a local minimum where F holds.

Polynomial Local Search (PLS) — and more generally, any Σ_1^b-definable function of a theory of bounded arithmetic — are special kinds of TFNP, Total NP Search, Problems:

Definition (Poljak-Turzík-Pudlák'82, JPY'88, Papadimitriou'94)

TFNP, the class of Total NP Functions is the set of polynomial time relations $R(x, y)$ such that $R(x, y)$ implies $|y| = |x|^{O(1)}$ and such that R is *total*, i.e., for all x, there exists y s.t. $R(x, y)$.

T^1_2 and PLS [B-Krajíček'94]

A *Polynomial Local Search* PLS is formalized in S^1_2 provided its feasible set, initial point function, neighborhood function, and cost function are Σ^{b}_1-defined (as polynomial time functions).

Theorem

T^1_2 *can prove that any (formalized)* PLS *problem is total.*

Proof: By Σ^{b}_1-minimization, T^1_2 can prove there is a minimum cost value c_0 satisfying

$$(\exists s \leq b(x))(F(x,s) \wedge c(x,s) = c_0).$$

Choosing s that realizes the cost c_0 gives either a solution to the PLS problem or a place where the PLS conditions are violated. $\square$

Theorem (B-Krajíček'94)

If $A \in \Sigma^b_1$ and $\mathrm{T}^1_2 \vdash (\forall x)(\exists y)A(x,y)$, then there is a PLS problem R such that T^1_2 proves

$$(\forall x)(\forall y)(R(x,y) \to A(x,(y)_1)).$$

If $A \in \Delta^b_1$, then can replace "$(y)_1$" with just "y".

This gives an exact complexity characterization of the $\forall\Sigma^b_1$-definable functions of T^1_2, in terms of PLS-computability. Namely:

Theorem

The Σ^b_1-definable (multi)functions of T^1_2 are precisely the projections of PLS functions.

Open: Can T^1_2 witness PLS problems using single-valued Σ^b_1-definable functions?

Theorem (Witnessing Lemma)

If $\Gamma \longrightarrow \Delta$ is a T_2^1-provable sequent of $\mathrm{ss}\Sigma_1^{\mathrm{b}}$ formulas with free variables $\vec{c}$, then there is a PLS problem $R(\langle \vec{c}, \vec{u} \rangle, v)$ so that T_2^1 proves

$$Wit_{\Gamma}(\vec{c}, \vec{u}) \wedge R(\langle \vec{c}, \vec{u} \rangle, v) \rightarrow Wit_{\Delta}(\vec{c}, v).$$

Proof idea: Use a free-cut free T_2^1-proof, proceed by induction on number of inferences in the proof. Arguments are similar to to what was used to prove the witnessing lemma for S_2^i ($i = 1$ case). Most cases just require closure of PLS under polynomial time operations. However, induction (Σ_1^{b}-IND inference) now requires exponentially long iteration: this is handled via the exponentially many possible cost values. □

Pause

Second order theories U_2^1 and V_2^1

For polynomial space and exponential time

We now consider theories of bounded arithmetic formulated in a second-order language.

- Second-order variables $X, Y, Z, \ldots$ or $\alpha, \beta, \gamma, \ldots$. These range over sets of integers.
- Viewed computationally, such an X can be viewed as an oracle.
- Notation: $t \in X$ is usually written as $X(t)$.
- Second-order variables implicitly have polynomial bounds on their members. This corresponds to the fact that there is a polynomial upper bound on the size of oracle queries to X.

Relativized versions of S_2^i and T_2^i

Definition ($\Sigma_i^{\mathrm{b}}(\alpha)$ and $\Pi_i^{\mathrm{b}}(\alpha)$)

$\Sigma_i^{\mathrm{b}}(\alpha)$ and $\Pi_i^{\mathrm{b}}(\alpha)$ are defined exactly like Σ_i^{b} and Π_i^{b} but now allowing atomic formulas $\alpha(t)$.

Definition

- $\mathrm{S}_2^i(\alpha)$ is: $\mathrm{BASIC} + \Sigma_i^{\mathrm{b}}(\alpha)$-PIND.
- $\mathrm{T}_2^i(\alpha)$ is: $\mathrm{BASIC} + \Sigma_i^{\mathrm{b}}(\alpha)$-IND.

$\mathrm{S}_2(\alpha) = \mathrm{T}_2(\alpha) = \cup_i \mathrm{T}_2^i(\alpha)$.

Theorem

- *The $\Sigma_1^{\mathrm{b}}(\alpha)$-definable functions of $\mathrm{S}_2^1(\alpha)$ are precisely the functions in P^α (so α is an oracle).*
- *The $\Sigma_1^{\mathrm{b}}(\alpha)$-definable functions of $\mathrm{T}_2^1(\alpha)$ are precisely the projections of PLS^α functions.*

A Hierarchy of Second-Order Formulas.

Definition (B'85)

- The $\Sigma_0^{1,\mathrm{b}} = \Pi_0^{1,\mathrm{b}}$ formulas are the formulas with bounded first order quantifiers, but no unbounded quantifiers and no second-order quantifiers.
- (For $i \geq 0$.) The class of $\Sigma_{i+1}^{1,\mathrm{b}}$ contains the formulas of the form $(\exists \vec{X})A(\vec{X})$ for A in $\Pi_i^{1,\mathrm{b}}$. We also close under conjunction and disjunction.
- The class of $\Pi_{i+1}^{1,\mathrm{b}}$-formulas is defined dually.

Informally: We count second-order quantifiers, disregard first-order quantifiers, and disallow unbounded quantifiers.

Remark: The $\Sigma_1^{1,\mathrm{b}}$-formulas define exactly the predicates in $\mathrm{NEXPTIME}$ (nondeterministic exponential time).

The theories U_2^1 and V_2^1

Definition (B'85)

- U_2^1 is $\mathrm{BASIC} + \Sigma_0^{1,\mathrm{b}}$-CA $+ \Sigma_1^{1,\mathrm{b}}$-PIND.
- V_2^1 is $\mathrm{BASIC} + \Sigma_0^{1,\mathrm{b}}$-CA $+ \Sigma_1^{1,\mathrm{b}}$-IND.

where $\Sigma_0^{1,\mathrm{b}}$-CA (Comprehension on bounded formulas) is

$$(\exists \alpha)[(\forall x)(\alpha(x) \leftrightarrow A(x, \vec{y}, \vec{\beta}))],$$

for all $\Sigma_0^{1,\mathrm{b}}$-formulas $A(x, \vec{y}, \vec{\beta})$.

Theorem (B'85)

The $\Sigma_1^{1,\mathrm{b}}$-definable functions

- *of U_2^1 are precisely the PSPACE-functions,*
- *of V_2^1 are precisely the $\mathrm{EXPTIME}$-functions.*

Summary of theories above

Theory	Axioms	Definable functions	Definability type
S^1_2	Σ^{b}_1-PIND	Poly. time (P)	Σ^{b}_1-definable
T^1_2	Σ^{b}_1-IND	Poly. Local Search (PLS)	Σ^{b}_1-definable
U^1_2	$\Sigma^{1,\mathrm{b}}_1$-PIND	PSPACE	$\Sigma^{1,\mathrm{b}}_1$-definable
V^1_2	$\Sigma^{1,\mathrm{b}}_1$-IND	$\mathrm{EXPTIME}$	$\Sigma^{1,\mathrm{b}}_1$-definable
S^i_2	Σ^{b}_i-PIND	$\mathrm{P}^{\Sigma^{\mathrm{b}}_{i-1}}$	Σ^{b}_i-definable
T^i_2	Σ^{b}_i-IND	$\mathrm{PLS}^{\Sigma^{\mathrm{b}}_{i-1}}$	Σ^{b}_i-definable

S^{i+1}_2 and T^i_2 have the same Σ^{b}_i-definable functions and the same $\Sigma^{\mathrm{b}}_{i+1}$-definable functions.

Second order theories VNC^1, VL and VNL

For ALogTime, L and NL

Weak second-order theories for weaker complexity
[Takeuti'91, Ignjatovic'95, Zambella'96, ..., Cook-Nguyen'10]
These second-order theories use
(a) first-order objects playing the role of sharply bounded objects,
(b) second-order objects playing the role of inputs and outputs.
Base theory V^0 has comprehension and induction for bounded first-order formulas (with second order free variables).

Theories for $\mathrm{ALogTime}$ (uniform NC^1): [CT, A, CM, CN]

- Complexity class NC^1 - properties expressible by polynomial size Boolean formulas.
- VNC^1 - is V^0 plus axioms asserting the totality of the Boolean Formula Value Problem or log-bounded tree recursion. These are in NC^1 [B] and complete for NC^1.
- Provably total functions are precisely the functions of polynomial growth rate with NC^1 bit graph.

Theories for L (log space) [Z, P, CN]

- VL - is V^0 plus axioms asserting the totality of log-bounded recursion.
- Provably total functions are precisely the log-space computable functions.

Theories for NL (nondeterministic log space) [CK, P, CN]

- VNL - is V^0 plus axioms asserting the existence of a distance predicate for graph reachability.
- Provably total functions are precisely the polynomial growth rate functions with NL bit graph.

End of third part!

Fourth part will discuss: Translations to propositional proofs.