Propositional proof systems and bounded arithmetic for logspace and nondeterministic logspace

Sam Buss
U.C. San Diego

Virtual Seminar
Proof Society
proofsociety.org
October 7, 2020

includes joint work with Anupam Das and Alexander Knop

Setting:

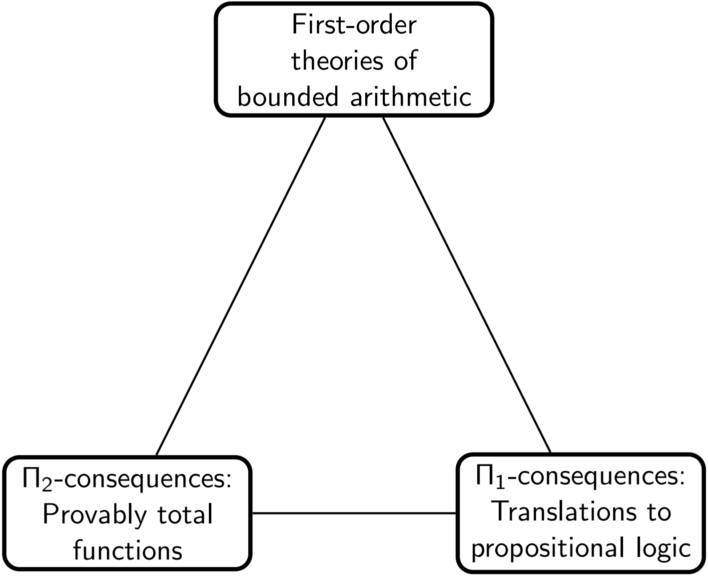
- Formal theories of weak fragments of Peano arithmetic
  - First- and second-order theories of bounded arithmetic
- $\forall\exists$ consequences: Provably total functions
  - Computational complexity characterizations
- $\forall$ consequences: Universal statements
  - Cook/Paris-Wilkie translation to propositional logic

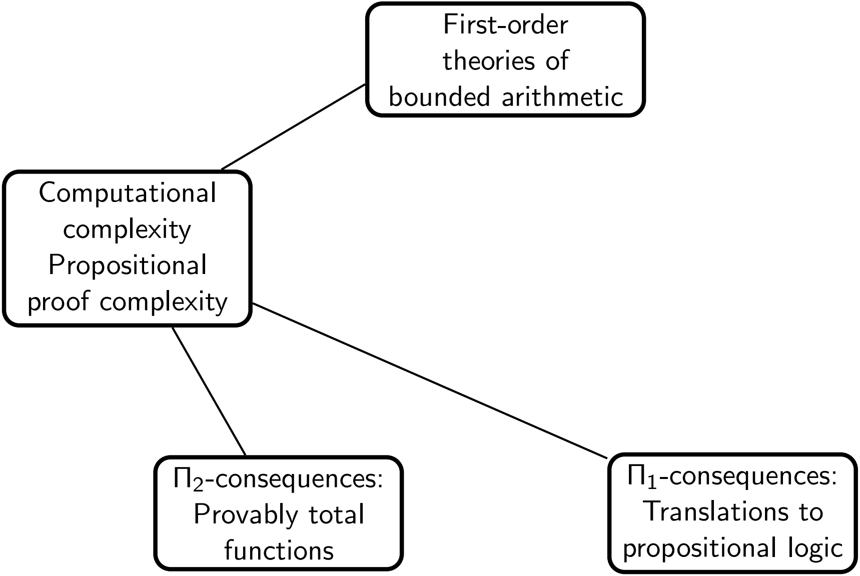
Underlying philosophy:

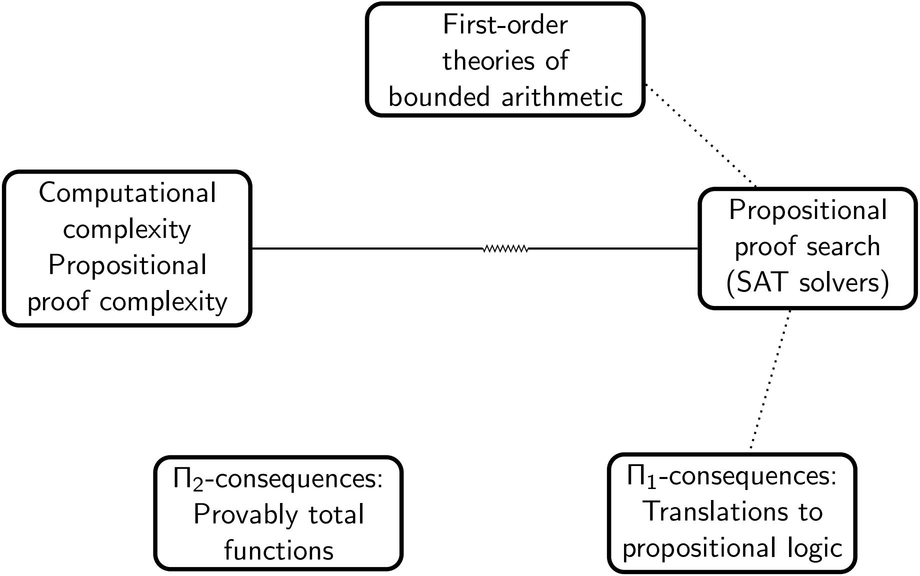
- A feasibly constructive proof that a function is total should provide a feasible method to compute it.
- A feasibly constructive proof of a universal statement should provide a feasible method to verify any given instance.

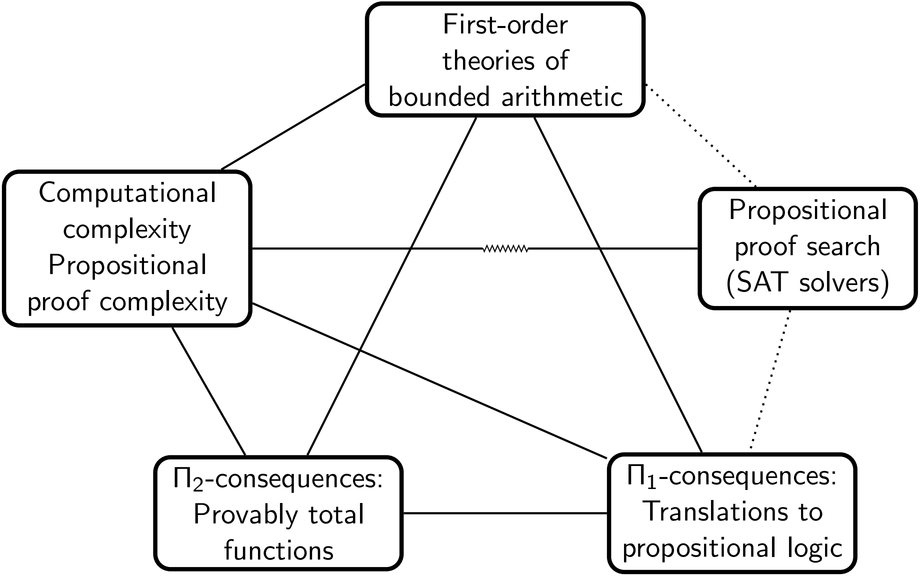
This talk (work-in-progress)

- Propositional and second-order systems for logspace and non-deterministic log space.


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


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


First-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 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 theory S^1_2 of arithmetic:

- Terms have polynomial growth rate (smash, $\#$, is used).
- Bounded quantifiers $\forall x \le t$, $\exists x \le t$.
- Sharply bounded quantifiers $\forall x \le |t|$, $\exists x \le |t|$,

  bound x by *log* (or *length*) of t.
- Classes Σ^{b}_i and Π^{b}_i of formulas are defined by counting bounded quantifiers, ignoring sharply bounded quantifiers.
- Σ^{b}_1 formulas express exactly the NP predicates.

  Σ^{b}_i, Π^{b}_i - express exactly the predicates at the i-th level of the polynomial time hierarchy.
- S^1_2 has *polynomial induction* PIND, equivalently *length induction* (LIND), for Σ^{b}_1 formulas A (i.e., NP formulas):

$$A(0) \land (\forall x)(A(x) \to A(x+1)) \to (\forall x)A(|x|)$$

(1) Provably total functions of S^1_2:

- The $\forall\Sigma^b_1$-definable functions (aka: *provably total functions*) are precisely the polynomial time computable functions.

(2) Translation to propositional logic ("Cook translation")

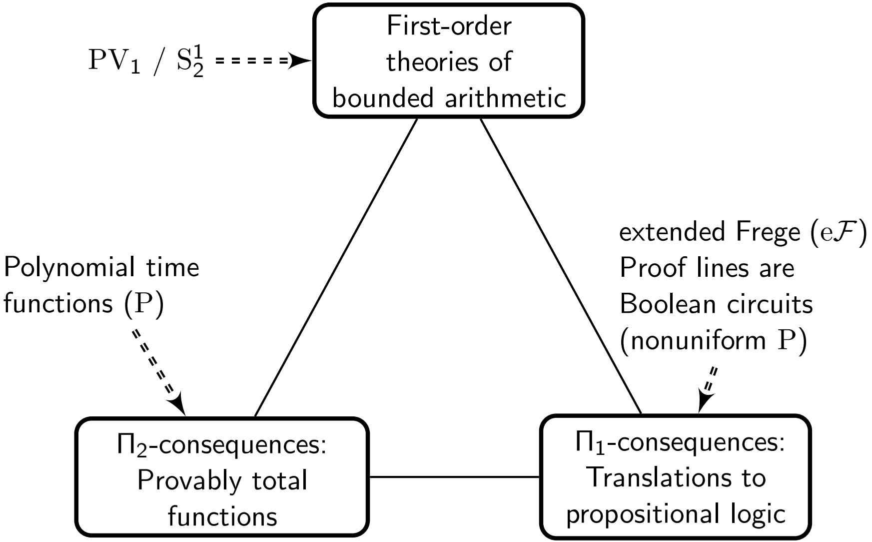
- Any polynomial identity ($\forall\Sigma^b_0$-property) provable in PV / S^1_2, has a natural translation to a family F of propositional formulas. These formulas have polynomial size extended Frege ($\mathrm{e}\mathcal{F}$) proofs.

(3) S^1_2 proves the consistency of $\mathrm{e}\mathcal{F}$. Conversely, any propositional proof system (p.p.s.) S^1_2 proves is consistent(provably) polynomially simulated by $\mathrm{e}\mathcal{F}$.

(4) Lines (formulas) in an $\mathrm{e}\mathcal{F}$ proof correspond to Boolean circuits. The circuit value problem is complete for P (polynomial time).

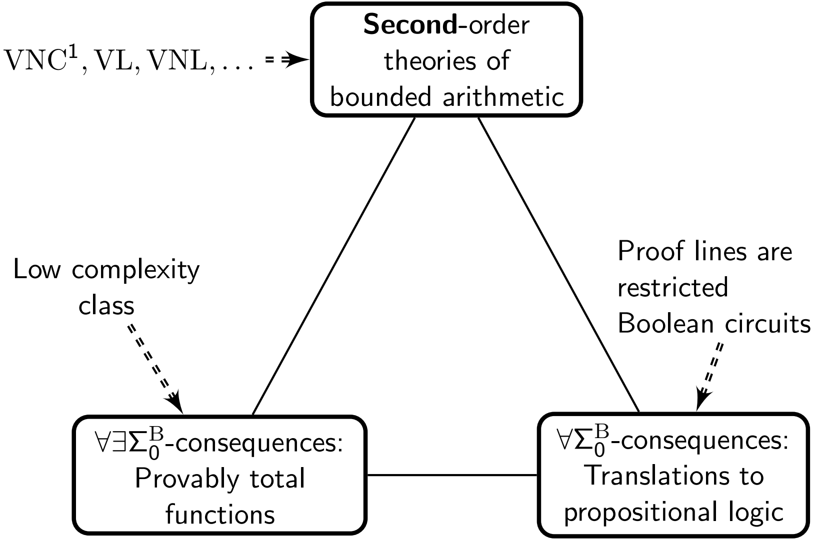
First-order theories work well for NC^2 and stronger classes.

E.g., for polynomial time:



For complexity classes below NC^2

[Clote-Takeuti; Zambella; Arai; Cook, Morioka, Perron, Kolokolova, Nguyen]



Weak second-order theories

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).

Syntax:

- First-order bounded quantifiers $\forall x{\le}t, \exists x{\le}t$ - range over small objects, namely "integers".
- Second-order quantifiers $\forall X, \exists X$ - range over large objects, namely (finite) sets of integers.
- $x \in Y$ or $Y(x)$ - set membership.
- Σ_0^{B} formulas have only bounded first-order quantifiers and no second-order quantifiers
- First-order arithmetic operations: $0, S, \mathrm{pd}, +, \cdot, \le, =$
- $|X|$ - maximal element in X (optional).

Axioms of base theory V_0:

- "Basic" axioms of first-order functions and $\leq$ and $=$.
- Boundedness: $\exists y\, \forall x (A(x) \rightarrow x \leq y)$.
- Minimization: $A(b) \rightarrow \exists x\, [A(x) \wedge \forall y < x. \neg A(y)]$.
- Σ_0^{B}-Comprehension $\exists X\, \forall y \leq a\, [X(y) \leftrightarrow \varphi(y)]$,
  for φ a Σ_0^{B}-formula (with parameters allowed).

Theories VL (for logspace) and VNL (for nondeterministic logspace) have additional axioms for totality of L and NL complete functions — *on next slides ...*

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

- VL - is V^0 plus the totality of log-bounded recursion.
- Provably total functions are precisely the log-space computable functions.
- Cook translation is a tree-like propositional proof system GL^* for Σ-$\mathrm{CNF}(2)$ formulas, a class of QBF formulas complete for log space. [J]

Log-bounded recursion axiom [Zambella]

$$
\begin{aligned}
&(\forall x{\le}a)(\exists y{\le}a)A(x,y) \to \exists X[X(0,0)\wedge \\
&\qquad (\forall i{\le}b)(\forall y{\le}a)(X(i,y) \to (\forall y'{<}y)\neg X(i,y')) \wedge \\
&\qquad (\forall i{<}b)(\exists y{\le}a)(\exists y'{\le}a)(X(i,y) \wedge X(i{+}1,y') \wedge A(y,y'))]
\end{aligned}
$$

Intuition: $A(x,y)$ denotes the (wlog deterministic) step for a path; defines a directed graph with out-degree ≥ 1..
$X(i,y)$ means y is the i-th vertex in the path.
The axiom asserts existence of a path of length b.
The predicate $X(i,y)$ is log-space complete.

A theory for $\mathbf{NL}$ (non-deterministic log space) [CK, P, CN]

- VNL - is V^0 plus the existence of a distance predicate for graph reachability.
- Provably total functions are precisely the polynomial growth rate functions with NL bit graph.
- Cook translation is a tree-like propositional proof system GNL^* for ΣKrom formulas, a class of QBF formulas complete for NL. [G]

Reachability/connectivity axiom [NC]

$$
\begin{aligned}
&(\exists X)[(\forall y{\le}a)(X(0,y) \leftrightarrow y=0) \wedge \\
&\qquad (\forall y{\le}a)(\forall i{<}b)[\,X(i{+}1,y) \leftrightarrow \\
&\qquad\qquad\qquad [\,X(i,y) \vee (\exists y'{<}a)(X(i,y') \wedge A(y',y))\,]\,]\,]
\end{aligned}
$$

Intuition: $A(x,y)$ denotes a possible step for a path.
$X(i,y)$ means y is reachable from 0 in $\le i$ steps.
The predicate $X(i,y)$ is NL complete.

Formal Theory	Propositional Proof System	Total Functions	
PV, S^1_2, VPV	$\mathrm{e}\mathcal{F}$, G^*_1	P	[C, B, CN]
T^1_2, S^2_2	G_1, G^*_2	$\le_{1\text{-}1}(\mathrm{PLS})$	[B, KP, KT, BK]
T^2_2, S^3_2	G_2, G^*_3	$\le_{1\text{-}1}(\mathrm{CPLS})$	[B, KP, KT, KST]
T^i_2, S^{i+1}_2	G_i, G^*_{i+1}	$\le_{1\text{-}1}(\mathrm{LLI}_i)$	[B, KP, KT, KNT]
PSA, U^1_2, W^1_1	QBF	*PSPACE*	[D, B, S]
V^1_2	**	*EXPTIME*	[B]
VNC^1	Frege ($\mathcal{F}$)	$\mathrm{ALogTime}$	[CT, A; CM, CN]
VL	GL^*	L	[Z, P, CN]
VNL	GNL^*	NL	[CK, P, CN]

PV, PSA - equational theories.
S^i_2, T^i_2 - first order
$\mathrm{U}^1_2, \mathrm{V}^1_2$, $\mathrm{VNC}^1, \mathrm{VL}, \mathrm{VNL}, \mathrm{VPV}$ - second order

Formal Theory	Propositional Proof System	Total Functions	
PV, S^1_2, VPV	$\mathrm{e}\mathcal{F}$, G^*_1	P	[C, B, CN]
T^1_2, S^2_2	G_1, G^*_2	$\leq_{1\text{-}1}(\mathrm{PLS})$	[B, KP, KT, BK]
T^2_2, S^3_2	G_2, G^*_3	$\leq_{1\text{-}1}(\mathrm{CPLS})$	[B, KP, KT, KST]
T^i_2, S^{i+1}_2	G_i, G^*_{i+1}	$\leq_{1\text{-}1}(\mathrm{LLI}_i)$	[B, KP, KT, KNT]
PSA, U^1_2, W^1_1	QBF	*PSPACE*	[D, B, S]
V^1_2	**	*EXPTIME*	[B]
VNC^1	Frege ($\mathcal{F}$)	$\mathrm{ALogTime}$	[CT, A; CM, CN]
VL	GL^*	L	[Z, P, CN]
VNL	GNL^*	NL	[CK, P, CN]

Using Cook translation to propositional proof systems (p.p.s.'s)

$\mathcal{F}, \mathrm{e}\mathcal{F}$ - Frege and extended Frege.

G_i, QBF - quantified propositional logics.

Starred (*) propositional systems are tree-like.

Formal Theory	Propositional Proof System	Total Functions	
PV, S_2^1, VPV	$\mathrm{e}\mathcal{F}$, G_1^*	P	[C, B, CN]
T_2^1, S_2^2	G_1, G_2^*	$\leq_{1\text{-}1}(\mathrm{PLS})$	[B, KP, KT, BK]
T_2^2, S_2^3	G_2, G_3^*	$\leq_{1\text{-}1}(\mathrm{CPLS})$	[B, KP, KT, KST]
T_2^i, S_2^{i+1}	G_i, G_{i+1}^*	$\leq_{1\text{-}1}(\mathrm{LLI}_i)$	[B, KP, KT, KNT]
PSA, U_2^1, W_1^1	QBF	*PSPACE*	[D, B, S]
V_2^1	**	*EXPTIME*	[B]
VNC^1	Frege ($\mathcal{F}$)	$\mathrm{ALogTime}$	[CT, A; CM, CN]
VL	GL^*	L	[Z, P, CN]
VNL	GNL^*	NL	[CK, P, CN]

PLS = Polynomial local search [JPY]

CPLS = "Colored" PLS [ST]

LLI = Linear local improvement

New Propositional Theories for L and NL [B-D-K]

The propositional theories GL^* and GNL^* are hard to work with since they are only indirectly connected with (non-uniform) log space and non-deterministic log space.

[Buss-Das-Knop'20]: Theories eLDT and eLNDT for propositional logic acting on

- Branching programs (eDT formulas), or
- Non-deterministic branching programs (eNDT formulas).

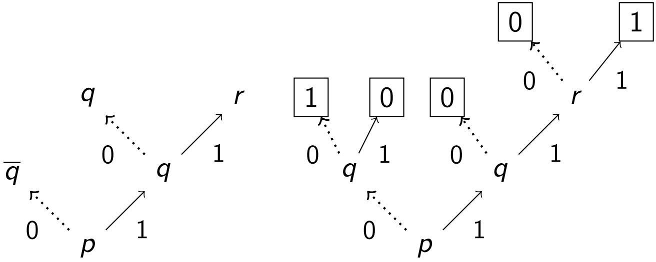
[Similar to a suggestion of Cook, but not using Prover-Liar games.]

Nomenclature: "DT" means "Decision Tree". An "eDT" ("extension Decision Tree") allows also extension variables, which converts the decision tree into a decision DAG, i.e., into a Branching Program (BP).

DT formulas

- **Atomic DT formulas:** p, $\overline{p}$, optionally 0 and 1.
- **Decision (or case/if-then-else) connective:** $(\varphi p \psi)$.
  Meaning "if p then ψ else φ" or "case(p,ψ,φ)".

Example:



These represent the equivalent formulas

$$\overline{q}\, p\, (q\, q\, r), \text{ and } (1q0)\, p\, (0\, q\, (0r1)).$$

Gentzen-style sequent calculus for DT/eDT formulas:

- Initial sequents, weak inferences, cut rule, and
- Decision connective rules:

$$\textit{dec-l:}\ \frac{A, \Gamma \longrightarrow \Delta, p \qquad p, B, \Gamma \longrightarrow \Delta}{(ApB), \Gamma \longrightarrow \Delta}$$

$$\textit{dec-r:}\ \frac{\Gamma \longrightarrow \Delta, A, p \qquad p, \Gamma \longrightarrow \Delta, B}{\Gamma \longrightarrow \Delta, (ApB)}$$

eDT - Extension DT formulas

Extension DT (eDT) formulas. Allow also introducing new extension variable, e, with a defining equation $e \leftrightarrow \varphi$.

eDT formulas, together with their defining equations express exactly (deterministic) branching programs.

Extension variables e can be used — unnegated — as atomic formulas, but cannot be used as decision literals.
The new initial sequents are $e \longrightarrow \varphi$ and $\varphi \longrightarrow e$.

- An eLDT proof is a sequence of sequents of eDT formulas inferred using the sequent calculus rules.
- The eDT formulas are used in conjunction with the defining equations for extension variables.
- Evaluation of eDT formulas is log-space complete. Thus, each line of an eLDT proof expresses a (non-uniform) logspace property.

Theorem ([BDK] - work in progress)

- *The* $\forall\Sigma_0^{\mathrm{B}}$*-consequences of* VL *have natural propositional translations which have polynomial size* eLDT*-proofs.*
- VL *can prove the consistency of* eLDT *proofs.*
- *Any propositional proof system which is* VL*-provably sound is p-simulated by* eLDT*.*

eLNDT for non-deterministic logspace

- NDT and eNDT formulas are defined exactly like DT and eDT formulas, but allowing also the connective $\vee$ (disjunction). Now it is important that extension variables can not be negated!
- An NDT formula expresses a "nondeterministic decision tree".
- An eNDT formula (together with defining equations for extension variables) expresses a "nondeterministic branching program".
- Evaluation of eNDT formulas (nondeterministic branching programs) is complete for non-deterministic logspace.
- An eLNDT proof consists of sequents of eNDT formulas.
- The permitted rules of inference are the inference rules for eLDT proofs (including the decision rule), plus the usual Gentzen $\vee$:left and $\vee$:right rules.

Theorem ([BDK] - work in progress)

- *The* $\forall\Sigma_0^{\mathrm{B}}$*-consequences of* VNL *have natural propositional translations which have polynomial size* eLNDT*-proofs.*
- VNL *can prove the consistency of* eLNDT *proofs.*
- *Any propositional proof system which is* VNL*-provably sound is p-simulated by* eLDT*.*

This includes (re)proving the Immerman-Szelepcsényi theorem that $\mathrm{NL} = \mathrm{coNL}$ in VNL. C.f. [CK, P].

Thank you for virtual attendance!