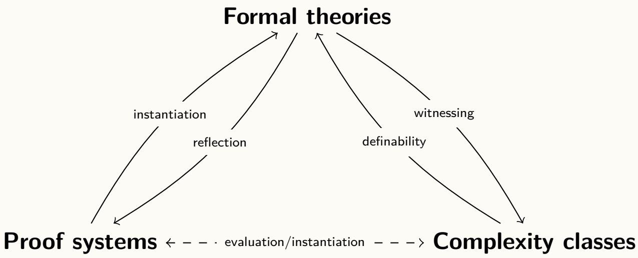
Proof complexity of systems of (non-deterministic) decision trees and branching programs

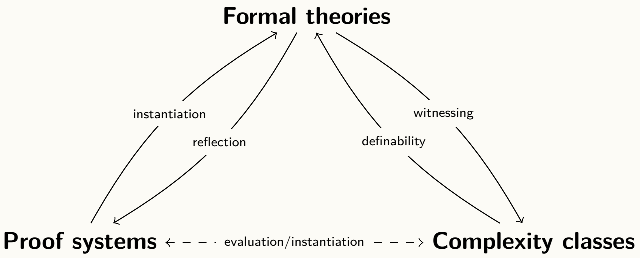
Authors:
Sam Buss, Anupam Das, Alexander Knop

Institute:
UC San Diego

The Bounded Arithmetic Correspondence

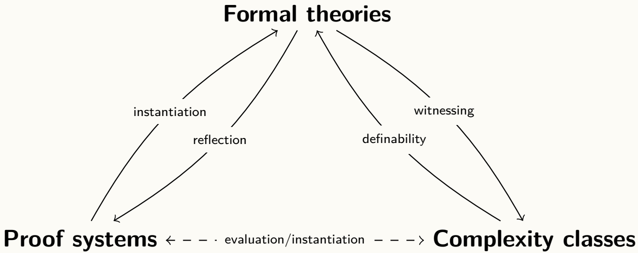


The Bounded Arithmetic Correspondence



A proof theoretic version of other “uniform-nonuniform” correspondences.

The Bounded Arithmetic Correspondence



A proof theoretic version of other “uniform-nonuniform” correspondences. E.g., the bottom dashed arrow, for **P**, is the correspondence between **P** machines and polynomial-size circuits.

Theories and Proof Systems

Let us illustrate this relation on the triplet extended Frege, S^1_2, and class $\mathbf{P}$.

Theories and Proof Systems

Let us illustrate this relation on the triplet extended Frege, S^1_2, and class **P**.
The vocabulary for S^1_2 is $\mathcal{L}_{\mathsf{S}_2} = [0, S, +, \cdot, \#, |x|, \left\lceil \frac{1}{2}x \right\rceil; =, \leq]$.

Theories and Proof Systems

Let us illustrate this relation on the triplet extended Frege, S^1_2, and class **P**. The vocabulary for S^1_2 is $\mathcal{L}_{\mathrm{S}_2} = [0, S, +, \cdot, \#, |x|, \left\lceil \frac{1}{2}x \right\rceil ; =, \leq]$. The axioms of S^1_2 are standard axioms of these operations and Σ^b_1-IND axiom, which says that

Theories and Proof Systems

Let us illustrate this relation on the triplet extended Frege, S^1_2, and class **P**. The vocabulary for S^1_2 is $\mathcal{L}_{\mathrm{S}_2} = [0, S, +, \cdot, \#, |x|, \left\lceil \frac{1}{2}x \right\rceil; =, \leq]$. The axioms of S^1_2 are standard axioms of these operations and Σ^b_1-IND axiom, which says that

$$(\phi(0) \wedge \forall x\, (\phi(x) \to \phi(x+1))) \to \forall z\, \phi(|z|).$$

Theories and Proof Systems

Let us illustrate this relation on the triplet extended Frege, S^1_2, and class **P**. The vocabulary for S^1_2 is $\mathcal{L}_{S_2} = [0, S, +, \cdot, \#, |x|, \left\lceil \frac{1}{2}x \right\rceil; =, \leq]$. The axioms of S^1_2 are standard axioms of these operations and Σ^b_1-IND axiom, which says that

$$(\phi(0) \wedge \forall x\, (\phi(x) \to \phi(x+1))) \to \forall z\, \phi(|z|).$$

Let $\phi(x)$ be a Σ^b_0. Then we can write, in a natural way, a propositional formula $[\![\phi]\!]_{n,\alpha}$ on the variables x_1, ..., x_n saying that A is true (α is an assignment to all other free variables).

Theories and Proof Systems

Let us illustrate this relation on the triplet extended Frege, S_2^1, and class **P**. So it is possible to prove the following theorem.

THEOREM

If $\mathsf{S}_2^1 \vdash \forall x\, \phi(x)$, *then* $[\![\phi]\!]_{n,\alpha}$ *has a polynomial size proof in extended Frege.*

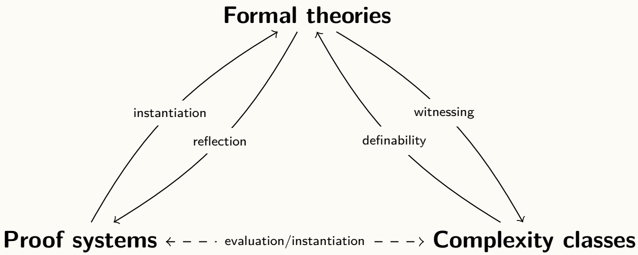
Theories and Proof Systems

Let us illustrate this relation on the triplet extended Frege, S^1_2, and class **P**.
So it is possible to prove the following theorem.

THEOREM

If $\mathsf{S}^1_2 \vdash \forall x\, \phi(x)$*, then* $[\![\phi]\!]_{n,\alpha}$ *has a polynomial size proof in extended Frege. Moreover,* S^1_2 *proves the reflection principle for extended Frege.*

The Bounded Arithmetic Correspondence



Theories and Complexity Classes

DEFINITION

A function $f : \mathbb{N} \to \mathbb{N}$ is Σ_1^b-definable by a theory R iff there is a Σ_1^b formula $A(x, y)$ such that

- $R \vdash \forall x\, \exists y \leq t\ A(x, y)$ for some term t,
- $R \vdash \forall x, y_1, y_2\ (A(x, y_1) \wedge A(x, y_2) \to y_1 = y_2)$, and
- A defines the graph of f.

Theories and Complexity Classes

DEFINITION

A function $f : \mathbb{N} \to \mathbb{N}$ is Σ^b_1-definable by a theory R iff 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,
- $R \vdash \forall x, y_1, y_2\; (A(x, y_1) \wedge A(x, y_2) \to y_1 = y_2)$, and
- A defines the graph of f.

THEOREM

S^1_2 *can* Σ^b_1*-define any polynomial time function.*

Theories and Complexity Classes

DEFINITION

A function $f : \mathbb{N} \to \mathbb{N}$ is Σ_1^b-definable by a theory R iff there is a Σ_1^b formula $A(x, y)$ such that

- $R \vdash \forall x\, \exists y \leq t\; A(x, y)$ for some term t,
- $R \vdash \forall x, y_1, y_2\; (A(x, y_1) \wedge A(x, y_2) \to y_1 = y_2)$, and
- A defines the graph of f.

THEOREM

S_2^1 can Σ_1^b-define any polynomial time function. Moreover, if f is Σ_1^b-definable by S_2^1, then f is polynomial time computable.

The Bounded Arithmetic Correspondence

Formal Theories	Propositional Proof Systems	Complexity Class	References
PV, S^1_2	e$\mathcal{F}$	**P**	[Coo75, Bus86]
PSA, U^1_2	G	**PSPACE**	[Dow78, Bus86]
T^i_2, S^{i+1}_2	G_i, G^*_{i+1}	$\mathbf{P}^{\Sigma^p_i}$	[KP90, KT92, Bus86]
VNC^0	$\mathcal{F}$	**ALogTime**	[CM05, CN10, Ara00]
VL	GL^*	**L**	[Per05, CN10]
VNL	GNL^*	**NL**	[Per09, CN10]

Previous Works

Proof systems corresponding to **L** and **NL** have been considered in the past:

- Perron gives systems based on logical characterisations of **L** and **NL**, namely CNF(2) and ΣKrom respectively. [Per05, Per09]
- Cook gives a game-theoretic system for **L** based on *branching programs*. (unpublished)

Previous Works

Proof systems corresponding to **L** and **NL** have been considered in the past:

- Perron gives systems based on logical characterisations of **L** and **NL**, namely CNF(2) and ΣKrom respectively. [Per05, Per09]
- Cook gives a game-theoretic system for **L** based on *branching programs*. (unpublished)

Can we achieve a similar correspondence for (**N**)**L** through a natural nonuniform model for (**N**)**L**, like for **ALogTime** and **P**?

Previous Works

Proof systems corresponding to **L** and **NL** have been considered in the past:

- Perron gives systems based on logical characterisations of **L** and **NL**, namely CNF(2) and ΣKrom respectively. [Per05, Per09]
- Cook gives a game-theoretic system for **L** based on *branching programs*. (unpublished)

Can we achieve a similar correspondence for (**N**)**L** through a natural nonuniform model for (**N**)**L**, like for **ALogTime** and **P**?

- Inspired by Cook's approach, we build a *bona fide* inference system based on **branching programs**.

Previous Works

Proof systems corresponding to **L** and **NL** have been considered in the past:

- Perron gives systems based on logical characterisations of **L** and **NL**, namely CNF(2) and ΣKrom respectively. [Per05, Per09]
- Cook gives a game-theoretic system for **L** based on *branching programs*. (unpublished)

Can we achieve a similar correspondence for (**N**)**L** through a natural nonuniform model for (**N**)**L**, like for **ALogTime** and **P**?

- Inspired by Cook's approach, we build a *bona fide* inference system based on **branching programs**.
- In particular, we treat decision trees, the tree-like branching programs, and recover dag-like ones by extension.

Branching Programs

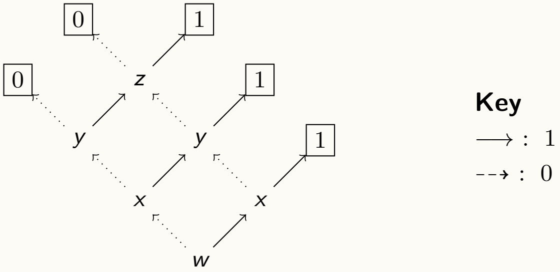
A *branching program* (BP) is a dag where:

- Each node is labelled by a propositional variable, 0 or 1;
- Each propositional node has two outgoing edges, labelled 0 and 1 respectively.

Branching Programs

A *branching program* (BP) is a dag where:

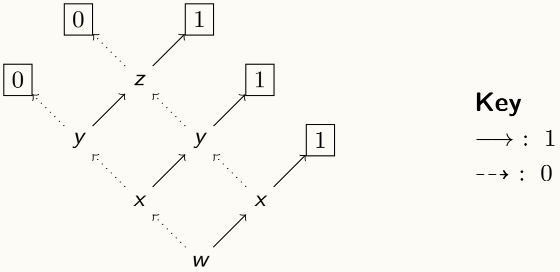
- Each node is labelled by a propositional variable, 0 or 1;
- Each propositional node has two outgoing edges, labelled 0 and 1 respectively.



Branching Programs

A *branching program* (BP) is a dag where:

- Each node is labelled by a propositional variable, 0 or 1;
- Each propositional node has two outgoing edges, labelled 0 and 1 respectively.



Can also consider *nondeterministic branching programs* (NBPs) and tree-like ones, *decision trees* (DTs) or both (NDTs).

A proof system for tree-like programs

Decision Tree (DT) formulas are built using a single "case" connective for literals:

$$ApB = \text{if } p \text{ then } B \text{ else } A$$

A proof system for tree-like programs

Decision Tree (DT) formulas are built using a single "case" connective for literals:

$$ApB = \text{if } p \text{ then } B \text{ else } A$$

Nondeterministic decision trees (NDTs) are obtained by allowing formulas to use disjunction, $\vee$.

A proof system for tree-like programs

Decision Tree (DT) formulas are built using a single "case" connective for literals:

$$ApB = \text{if } p \text{ then } B \text{ else } A$$

Nondeterministic decision trees (NDTs) are obtained by allowing formulas to use disjunction, $\vee$.

The system LDT is a sequent calculus with standard structural rules and the following logical rules for DT formulas:

A proof system for tree-like programs

Decision Tree (DT) formulas are built using a single "case" connective for literals:

$$ApB = \text{if } p \text{ then } B \text{ else } A$$

Nondeterministic decision trees (NDTs) are obtained by allowing formulas to use disjunction, $\vee$.

The system LDT is a sequent calculus with standard structural rules and the following logical rules for DT formulas:

$$\text{dec-}l\,\frac{\Gamma, A \to p, \Delta \quad \Gamma, p, B \to \Delta}{\Gamma, ApB \to \Delta} \qquad \text{dec-}r\,\frac{\Gamma \to A, p, \Delta \quad \Gamma, p \to B, \Delta}{\Gamma \to ApB, \Delta}$$

A proof system for tree-like programs

Decision Tree (DT) formulas are built using a single "case" connective for literals:

$$ApB = \text{if } p \text{ then } B \text{ else } A$$

Nondeterministic decision trees (NDTs) are obtained by allowing formulas to use disjunction, $\vee$.

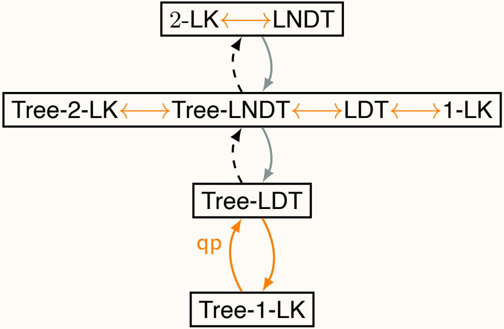
The system LDT is a sequent calculus with standard structural rules and the following logical rules for DT formulas:

$$\text{dec-}l\,\frac{\Gamma, A \rightarrow p, \Delta \quad \Gamma, p, B \rightarrow \Delta}{\Gamma, ApB \rightarrow \Delta} \qquad \text{dec-}r\,\frac{\Gamma \rightarrow A, p, \Delta \quad \Gamma, p \rightarrow B, \Delta}{\Gamma \rightarrow ApB, \Delta}$$

The system LNDT extends LDT by standard rules for $\vee$:

$$\vee\text{-}l\,\frac{\Gamma, A \rightarrow \Delta \quad \Gamma, B \rightarrow \Delta}{\Gamma, A \vee B \rightarrow \Delta} \qquad \vee\text{-}r\,\frac{\Gamma \rightarrow A, B, \Delta}{\Gamma \rightarrow A \vee B, \Delta}$$

L(N)DT Proofs



Key

$\rightarrow$: p-simulates

$\underset{\text{qp}}{\rightarrow}$: qp-simulates

$\dashrightarrow$: separated

orange : our results

gray : immediate

eL(N)DT proofs

- *Dags* cannot be expressed naturally as formulas, which are just labelled trees.

eL(N)DT proofs

- *Dags* cannot be expressed naturally as formulas, which are just labelled trees.
- They are typically handled in proof complexity via extension variables, which allow us to encode local properties of a graph.

eL(N)DT proofs

- *Dags* cannot be expressed naturally as formulas, which are just labelled trees.
- They are typically handled in proof complexity via extension variables, which allow us to encode local properties of a graph. (The very same idea allows extended Frege to reason about Boolean circuits instead of just formulas.)

eL(N)DT proofs

- *Dags* cannot be expressed naturally as formulas, which are just labelled trees.
- They are typically handled in proof complexity via extension variables, which allow us to encode local properties of a graph. (The very same idea allows extended Frege to reason about Boolean circuits instead of just formulas.)
- We allow *extension variables* e_1, e_2, etc. to be formulas, but importantly may not occur as decision literals.

eL(N)DT proofs

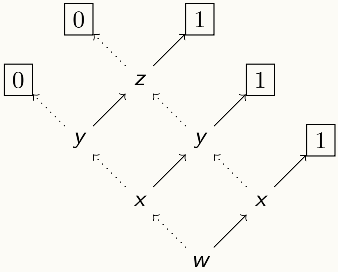
- *Dags* cannot be expressed naturally as formulas, which are just labelled trees.
- They are typically handled in proof complexity via extension variables, which allow us to encode local properties of a graph. (The very same idea allows extended Frege to reason about Boolean circuits instead of just formulas.)
- We allow *extension variables* e_1, e_2, etc. to be formulas, but importantly may not occur as decision literals.
- **Intuition:** the variables e_i are used to *name* subprograms, but querying whole subprograms amounts to the power of Boolean circuits.

eL(N)DT proofs

- *Dags* cannot be expressed naturally as formulas, which are just labelled trees.
- They are typically handled in proof complexity via extension variables, which allow us to encode local properties of a graph. (The very same idea allows extended Frege to reason about Boolean circuits instead of just formulas.)
- We allow *extension variables* e_1, e_2, etc. to be formulas, but importantly may not occur as decision literals.
- **Intuition:** the variables e_i are used to *name* subprograms, but querying whole subprograms amounts to the power of Boolean circuits.

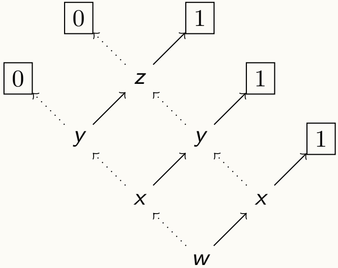
A proof of eLDT or eLNDT is just like that of LDT or LNDT, but comes equipped with a set of axioms of the form $e_n \leftrightarrow A_n(e_i)_{i<n}$. The conclusion of such a proof must not contain extension variables.

Example



$$
\begin{aligned}
e_{00} &\leftrightarrow e_{10}\,w\,e_{11}\\
e_{10} &\leftrightarrow e_{20}\,x\,e_{21}\\
e_{11} &\leftrightarrow e_{21}\,x\,1\\
e_{20} &\leftrightarrow 0\,y\,e_{31}\\
e_{21} &\leftrightarrow e_{31}\,y\,1\\
e_{31} &\leftrightarrow 0\,z\,1
\end{aligned}
$$

Example



$$
\begin{aligned}
e_{00} &\leftrightarrow e_{10}\, w e_{11} \\
e_{10} &\leftrightarrow e_{20}\, x e_{21} \\
e_{11} &\leftrightarrow e_{21}\, x 1 \\
e_{20} &\leftrightarrow 0 y e_{31} \\
e_{21} &\leftrightarrow e_{31}\, y 1 \\
e_{31} &\leftrightarrow 0 z 1
\end{aligned}
$$

- Here e_{ij} names the jth node, left-right, of the ith row, bottom-up.
- The entire program is now expressed by e_{00}.

The problem of equivalence

- One technicality, arising from the fact that a single branching program may be written in several ways with extension, is to prove the equivalence of *isomorphic* branching programs.

The problem of equivalence

- One technicality, arising from the fact that a single branching program may be written in several ways with extension, is to prove the equivalence of *isomorphic* branching programs.
- Works such as [Jeř04] propose to simply include axioms/rules for such situations, but this is undesirable as isomorphism is not known to be in **L**.

The problem of equivalence

- One technicality, arising from the fact that a single branching program may be written in several ways with extension, is to prove the equivalence of *isomorphic* branching programs.
- Works such as [Jeř04] propose to simply include axioms/rules for such situations, but this is undesirable as isomorphism is not known to be in **L**.
- Instead, we enforce that this equivalence must be carried out explicitly in proofs.

The problem of equivalence

- One technicality, arising from the fact that a single branching program may be written in several ways with extension, is to prove the equivalence of *isomorphic* branching programs.
- Works such as [Jeř04] propose to simply include axioms/rules for such situations, but this is undesirable as isomorphism is not known to be in **L**.
- Instead, we enforce that this equivalence must be carried out explicitly in proofs.

LEMMA

The equivalence of isomorphic (N)BPs has polynomial-size proofs in eL(N)DT.

The problem of equivalence

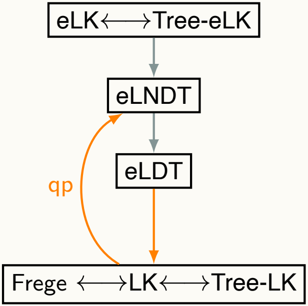
- One technicality, arising from the fact that a single branching program may be written in several ways with extension, is to prove the equivalence of *isomorphic* branching programs.
- Works such as [Jeř04] propose to simply include axioms/rules for such situations, but this is undesirable as isomorphism is not known to be in **L**.
- Instead, we enforce that this equivalence must be carried out explicitly in proofs.

LEMMA

The equivalence of isomorphic (N)BPs has polynomial-size proofs in eL(N)DT.

NB: these proofs are crucially dag-like!

Results



Key

$\rightarrow$: polynomially-simulates

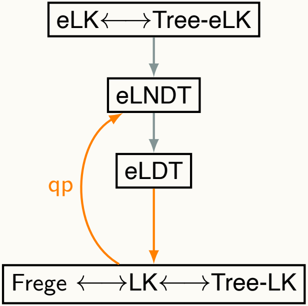
$\underset{\text{qp}}{\rightarrow}$: quasipolynomially-simulates

$\dashrightarrow$: exponentially separated from

orange : our results

gray : immediate

Results



Key

$\rightarrow$: polynomially-simulates

$\underset{qp}{\rightarrow}$: quasipolynomially-simulates

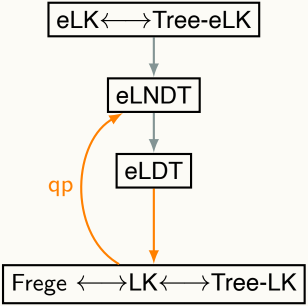
$\dashrightarrow$: exponentially separated from

orange : our results

gray : immediate

- Results follow by direct simulations, under equivalence of isomorphic (N)BPs.

Results



Key

$\rightarrow$: polynomially-simulates

$\underset{\text{qp}}{\rightarrow}$: quasipolynomially-simulates

$\dashrightarrow$: exponentially separated from

orange : our results

gray : immediate

- Results follow by direct simulations, under equivalence of isomorphic (N)BPs.
- We rely on Buss' qp-size formulas for st-connectivity and their small proofs in LK to evaluate NBPs and prove truth conditions. [Bus15].

VL and VNL

These theories are two-sorted theories.

VL and VNL

These theories are two-sorted theories. The vocabulary for both theories is $\mathcal{L}_{V^0} = [0, 1, \mathsf{pd}, +, \cdot; =, \leq, \in]$.

VL and VNL

These theories are two-sorted theories. The vocabulary for both theories is $\mathcal{L}_{V^0} = [0, 1, \mathsf{pd}, +, \cdot; =, \leq, \in]$. The axioms for the first-order objects are standards.

VL and VNL

These theories are two-sorted theories. The vocabulary for both theories is $\mathcal{L}_{V^0} = [0, 1, \mathsf{pd}, +, \cdot; =, \leq, \in]$. The axioms for the first-order objects are standards. In addition, we have boundedness, minimization, and Σ^b_0-comprehension:

- $\exists x \, \forall y \, (A(x) \to x \leq y)$.

VL and VNL

These theories are two-sorted theories. The vocabulary for both theories is $\mathcal{L}_{V^0} = [0, 1, \mathsf{pd}, +, \cdot; =, \leq, \in]$. The axioms for the first-order objects are standards. In addition, we have boundedness, minimization, and Σ^b_0-comprehension:

- $\exists x \, \forall y \, (A(x) \to x \leq y)$.
- $b \in A \to \big(\exists x \leq b \; A(x) \wedge (\forall y < x \, \neg A(y))\big)$.

VL and VNL

These theories are two-sorted theories. The vocabulary for both theories is $\mathcal{L}_{V^0} = [0, 1, \mathsf{pd}, +, \cdot; =, \leq, \in]$. The axioms for the first-order objects are standards. In addition, we have boundedness, minimization, and Σ_0^b-comprehension:

- $\exists x\, \forall y\, (A(x) \to x \leq y)$.
- $b \in A \to \big(\exists x \leq b\; A(x) \wedge (\forall y < x\; \neg A(y))\big)$.
- $\exists X\, \forall x \leq a\, (X(x) \leftrightarrow \phi(x))$, where ϕ is a Σ_0^b formula.

VL and VNL

These theories are two-sorted theories. The vocabulary for both theories is $\mathcal{L}_{V^0} = [0, 1, \mathsf{pd}, +, \cdot; =, \leq, \in]$. The axioms for the first-order objects are standards. In addition, we have boundedness, minimization, and Σ_0^b-comprehension:

- $\exists x \, \forall y \, (A(x) \to x \leq y)$.
- $b \in A \to \big(\exists x \leq b \; A(x) \wedge (\forall y < x \; \neg A(y))\big)$.
- $\exists X \, \forall x \leq a \; (X(x) \leftrightarrow \phi(x))$, where ϕ is a Σ_0^b formula.

VL and VNL

These theories are two-sorted theories. The vocabulary for both theories is $\mathcal{L}_{V^0} = [0, 1, \mathsf{pd}, +, \cdot; =, \leq, \in]$. The axioms for the first-order objects are standards. In addition, we have boundedness, minimization, and Σ_0^b-comprehension:

- $\exists x \, \forall y \, (A(x) \to x \leq y)$.
- $b \in A \to \big(\exists x \leq b \; A(x) \wedge (\forall y < x \; \neg A(y))\big)$.
- $\exists X \, \forall x \leq a \; (X(x) \leftrightarrow \phi(x))$, where ϕ is a Σ_0^b formula.

VL also has an axiom saying that if each vertex in a directed graph has out degree 1, then there is a path of length ℓ for any ℓ.

VL and VNL

These theories are two-sorted theories. The vocabulary for both theories is $\mathcal{L}_{V^0} = [0, 1, \mathsf{pd}, +, \cdot; =, \leq, \in]$. The axioms for the first-order objects are standards. In addition, we have boundedness, minimization, and Σ_0^b-comprehension:

- $\exists x\, \forall y\, (A(x) \to x \leq y)$.
- $b \in A \to \big(\exists x \leq b\; A(x) \wedge (\forall y < x\, \neg A(y))\big)$.
- $\exists X\, \forall x \leq a\, (X(x) \leftrightarrow \phi(x))$, where ϕ is a Σ_0^b formula.

VL also has an axiom saying that if each vertex in a directed graph has out degree 1, then there is a path of length ℓ for any ℓ.

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

VL and VNL

These theories are two-sorted theories.

VL and VNL

These theories are two-sorted theories. The vocabulary for both theories is $\mathcal{L}_{V^0} = [0, 1, \mathsf{pd}, +, \cdot; =, \leq, \in]$. The axioms for the first-order objects are standards. In addition, we have boundedness, minimization, and Σ_0^b-comprehension:

- $\exists x\, \forall y\, (A(x) \to x \leq y)$.
- $b \in A \to \big(\exists x \leq b\; A(x) \wedge (\forall y < x\, \neg A(y))\big)$.
- $\exists X\, \forall x \leq a\; (X(x) \leftrightarrow \phi(x))$, where ϕ is a Σ_0^b formula.

VL and VNL

These theories are two-sorted theories. The vocabulary for both theories is $\mathcal{L}_{V^0} = [0, 1, \mathsf{pd}, +, \cdot; =, \leq, \in]$. The axioms for the first-order objects are standards. In addition, we have boundedness, minimization, and Σ_0^b-comprehension:

- $\exists x \, \forall y \, (A(x) \to x \leq y)$.
- $b \in A \to \big(\exists x \leq b \; A(x) \land (\forall y < x \, \neg A(y))\big)$.
- $\exists X \, \forall x \leq a \; (X(x) \leftrightarrow \phi(x))$, where ϕ is a Σ_0^b formula.

VNL has an axiom saying that there is a function that gives distance from any fixed vertex.

$$\exists X \leq \langle a, a \rangle (\forall i \leq a \; X(0, i) \leftrightarrow (i = 0)) \land \\ \big(\forall w, x \leq a \; (X(x, w+1) \leftrightarrow [\exists y \leq a \; X(y, w) \land \phi(y, x)])\big)$$

Cook-style Translation

Let ϕ be a Σ_0^b formula with one free second-order variable X.

Cook-style Translation

Let ϕ be a Σ_0^b formula with one free second-order variable X. Let α be an assignment to all the first-order variables of ϕ.

Cook-style Translation

Let ϕ be a Σ_0^b formula with one free second-order variable X. Let α be an assignment to all the first-order variables of ϕ. Then $[\![\phi]\!]_{n,\alpha}$ is a propositional formula on the variables x_1, ..., x_n saying that A is true under the assignment α and for X such that $X(i)$ iff x_i is true.

Cook-style Translation

Let ϕ be a Σ_0^b formula with one free second-order variable X. Let α be an assignment to all the first-order variables of ϕ. Then $[\![\phi]\!]_{n,\alpha}$ is a propositional formula on the variables x_1, ..., x_n saying that A is true under the assignment α and for X such that $X(i)$ iff x_i is true.

THEOREM

- *If **VL** $\vdash \exists X\, \phi(X)$, then there is a eLDT proof of $[\![\phi]\!]_{n,\alpha}$.*
- *If **VNL** $\vdash \exists X\, \phi(X)$, then there is a eLNDT proof of $[\![\phi]\!]_{n,\alpha}$.*

Cook-style Translation

THEOREM

Let ϕ be a Σ_0^b formula.

- *If* **VL** $\vdash \exists X\, \phi(X)$, *then there is a eLDT proof of* $[\![\phi]\!]_{n,\alpha}$.
- *If* **VNL** $\vdash \exists X\, \phi(X)$, *then there is a eLNDT proof of* $[\![\phi]\!]_{n,\alpha}$.

The idea of the proof is to use structural induction over the proofs in VL and VNL.

Cook-style Translation

THEOREM

Let ϕ be a Σ_0^b formula.

- *If* $\mathbf{VL} \vdash \exists X\, \phi(X)$, *then there is a eLDT proof of* $[\![\phi]\!]_{n,\alpha}$.
- *If* $\mathbf{VNL} \vdash \exists X\, \phi(X)$, *then there is a eLNDT proof of* $[\![\phi]\!]_{n,\alpha}$.

The idea of the proof is to use structural induction over the proofs in VL and VNL. In other words, we are going to try to prove that

$$\frac{\Gamma'' \to \Delta'' \quad \Gamma \to \Delta}{\Gamma' \to \Delta'}$$

in V(N)L, then

$$\frac{[\![\Gamma'']\!]_{n,\alpha} \to [\![\Delta'']\!]_{n,\alpha} \quad [\![\Gamma]\!]_{n,\alpha} \to [\![\Delta]\!]_{n,\alpha}}{[\![\Gamma']\!]_{n,\alpha} \to [\![\Delta']\!]_{n,\alpha}}$$

in eL(N)DT.

Cook-style Translation

THEOREM

Let ϕ be a Σ^b_0 formula.

- *If* **VL** $\vdash \exists X\, \phi(X)$, *then there is a eLDT proof of* $[\![\phi]\!]_{n,\alpha}$.
- *If* **VNL** $\vdash \exists X\, \phi(X)$, *then there is a eLNDT proof of* $[\![\phi]\!]_{n,\alpha}$.

The idea of the proof is to use structural induction over the proofs in VL and VNL. In other words, we are going to try to prove that

$$\frac{\Gamma'' \to \Delta'' \quad \Gamma \to \Delta}{\Gamma' \to \Delta'}$$

in V(N)L, then

$$\frac{[\![\Gamma'']\!]_{n,\alpha} \to [\![\Delta'']\!]_{n,\alpha} \quad [\![\Gamma]\!]_{n,\alpha} \to [\![\Delta]\!]_{n,\alpha}}{[\![\Gamma']\!]_{n,\alpha} \to [\![\Delta']\!]_{n,\alpha}}$$

in eL(N)DT.

The problem of this approach is that the sequents my have Σ^b_1 formulas since the axiomatizations of VL and VNL have Σ^b_1 axioms.

Cook-style Translation

We create a theory T such that any Σ_0^b formula provable in VL (VNL) is provable in T; but T has a Σ_0^b axiomatization.

Cook-style Translation

We create a theory T such that any Σ_0^b formula provable in VL (VNL) is provable in T; but T has a Σ_0^b axiomatization.
To prove this we introduced predicate symbols instead of second-order objects guaranteed by the axioms of VL and VNL.

Cook-style Translation

We create a theory T such that any Σ_0^b formula provable in VL (VNL) is provable in T; but T has a Σ_0^b axiomatization.
To prove this we introduced predicate symbols instead of second-order objects guaranteed by the axioms of VL and VNL. It is clear that they are computable by eLDT and eLNDT, respectively. So we can extend the transformation to the formulas in T.

In case of VL this actually works, but in case of VNL there is a problem...

Cook-style Translation

We create a theory T such that any Σ_0^b formula provable in VL (VNL) is provable in T; but T has a Σ_0^b axiomatization.
To prove this we introduced predicate symbols instead of second-order objects guaranteed by the axioms of VL and VNL. It is clear that they are computable by eLDT and eLNDT, respectively. So we can extend the transformation to the formulas in T.

In case of VL this actually works, but in case of VNL there is a problem...
Negation of the reachability has no clean representation as a eLNDT.

Cook-style Translation

We create a theory T such that any Σ_0^b formula provable in VL (VNL) is provable in T; but T has a Σ_0^b axiomatization.
To prove this we introduced predicate symbols instead of second-order objects guaranteed by the axioms of VL and VNL. It is clear that they are computable by eLDT and eLNDT, respectively. So we can extend the transformation to the formulas in T.

In case of VL this actually works, but in case of VNL there is a problem...
Negation of the reachability has no clean representation as a eLNDT. To avoid this, we need to prove some analogue of Immerman–Szelepcsényi's theorem.

Cook-style Translation

Thank you!

References I

Toshiyasu Arai.
A bounded arithmetic AID for Frege systems.
Annals of Pure and Applied Logic, 103:155–199, 2000.

Samuel R. Buss.
Bounded Arithmetic.
Bibliopolis, Naples, Italy, 1986.
Revision of 1985 Princeton University Ph.D. thesis.

Sam Buss.
Quasipolynomial size proofs of the propositional pigeonhole principle.
Theoretical Computer Science, 576(C):77–84, 2015.

Stephen A. Cook and Tsuyoshi Morioka.
Quantified propositional calculus and a second-order theory for NC^1.
Archive for Mathematical Logic, 44:711–749, 2005.

References II

Stephen A. Cook and Phuong Nguyen.

Foundations of Proof Complexity: Bounded Arithmetic and Propositional Translations.

ASL and Cambridge University Press, 2010.

496 pages.

Stephen A. Cook.

Feasibly constructive proofs and the propositional calculus.

In *Proceedings of the Seventh Annual ACM Symposium on Theory of Computing*, pages 83–97. Association for Computing Machinery, 1975.

Martin Dowd.

Propositional representation of arithmetic proofs.

In *Proceedings of the 10th ACM Symposium on Theory of Computing (STOC)*, pages 246–252, 1978.

Emil Jeřábek.

Dual weak pigeonhole principle, Boolean complexity, and derandomization.

Annals of Pure and Applied Logic, 124:1–37, 2004.

References III

Jan Krajíček and Pavel Pudlák.
Quantified propositional calculi and fragments of bounded arithmetic.
Zeitschrift für Mathematische Logik und Grundlagen der Mathematik, 36:29–46, 1990.

Jan Krajíček and Gaisi Takeuti.
On induction-free provability.
Annals of Mathematics and Artificial Intelligence, pages 107–126, 1992.

Steven Perron.
A propositional proof system for log space.
In *Proc. 14th Annual Conf. Computer Science Logic (CSL)*, Springer Verlag Lecture Notes in Computer Science 3634, pages 509–524, 2005.

Steven Perron.
Power of Non-Uniformity in Proof Complexity.
PhD thesis, Department of Computer Science, University of Toronto, 2009.