Proof Mining in Bounded Arithmetic

Amir Akbar Tabatabai

Institute of Mathematics, Czech Academy of Sciences

amir.akbar@gmail.com

July 19, 2018

The Computational Content

In the arithmetical world, any bounded formula leads to its own canonical computational task whose solvability is guaranteed by the truth of the universal closure of the formula itself.

The Computational Content

In the arithmetical world, any bounded formula leads to its own canonical computational task whose solvability is guaranteed by the truth of the universal closure of the formula itself. This task is roughly defined as:

The computational content of A

Read the free variables, $\vec{x}$, as the parameters, and then alternately read the universal quantifiers of $A(\vec{x})$ to witness its existential quantifiers in a way that the resulting statement becomes true.

The Computational Content

In the arithmetical world, any bounded formula leads to its own canonical computational task whose solvability is guaranteed by the truth of the universal closure of the formula itself. This task is roughly defined as:

The computational content of A

Read the free variables, $\vec{x}$, as the parameters, and then alternately read the universal quantifiers of $A(\vec{x})$ to witness its existential quantifiers in a way that the resulting statement becomes true.

For instance consider the formula

$$A(x) = \forall y \leq t(x) \exists z \leq s(x) \forall w \leq r(x) B(x, y, z, w)$$

where B is a quantifier-free formula in the language of arithmetic and t, s and r are computable function symbols. Then the computational content of this sentence is the problem of reading x and $y \leq t(x)$ to find $z \leq s(x)$ such that for any $w \leq r(x)$, $B(x, y, z, w)$.

Proof Mining

- Clearly, there exists a simple brute-force algorithm to solve all these problems. The idea is simple: Since everything is bounded, it is just enough to check all the possibilities to find the needed witnesses.

Proof Mining

- Clearly, there exists a simple brute-force algorithm to solve all these problems. The idea is simple: Since everything is bounded, it is just enough to check all the possibilities to find the needed witnesses.
- The algorithm is based on the truth of $\forall \vec{x} A(\vec{x})$. But what if we also have a proof of $\forall \vec{x} A(\vec{x})$ in some theory T? Following Kreisel:

Proof Mining

- Clearly, there exists a simple brute-force algorithm to solve all these problems. The idea is simple: Since everything is bounded, it is just enough to check all the possibilities to find the needed witnesses.
- The algorithm is based on the truth of $\forall \vec{x} A(\vec{x})$. But what if we also have a proof of $\forall \vec{x} A(\vec{x})$ in some theory T? Following Kreisel:

Proof Mining

Does a T-proof of a bounded formula lead to a more clever algorithm than the mentioned blind brute-force?

Proof Mining

- Clearly, there exists a simple brute-force algorithm to solve all these problems. The idea is simple: Since everything is bounded, it is just enough to check all the possibilities to find the needed witnesses.
- The algorithm is based on the truth of $\forall \vec{x} A(\vec{x})$. But what if we also have a proof of $\forall \vec{x} A(\vec{x})$ in some theory T? Following Kreisel:

Proof Mining

Does a T-proof of a bounded formula lead to a more clever algorithm than the mentioned blind brute-force?

The answer is yes and this talk is devoted to show how. For this purpose, we will explain a general extraction technique applicable to any bounded theory of arithmetic. For more powerful unbounded theories like PA, there is a similar technique based on the ordinal analysis of the theory which we have to avoid here.

Reductions

Definition

Let A, B are two bounded formulas in the form

$$A = \forall y_1 \leq t_1(x) \exists y_2 \leq t_2(x) \ldots Q_n y_n \leq t_n(x) A_*(x, y_1, \ldots, y_n)$$

and

$$B = \forall z_1 \leq s_1(x) \exists z_2 \leq s_2(x) \ldots Q_n z_n \leq s_n(x) B_*(x, z_1, \ldots, z_n)$$

Then we say B is reducible to A and we write $A \geq B$ iff there exists a sequence of terms, alternately reading the universal variables of B to witness the universal quantifiers in A and then reading the existential variables in A to witness the existential variables in B. If the fact is provable in a base theory $\mathcal{B}$, the reduction is called $\mathcal{B}$-verifiable.

Reductions

Therefore, a reduction from B to A means a term-based coordinate change to reduce the computational task of B to the computational task of A.

Reductions

Therefore, a reduction from B to A means a term-based coordinate change to reduce the computational task of B to the computational task of A.

Let us illuminate the definition by one example:

Example

A reduction

$$[\forall y \leq t(x) \exists z \leq s(x) A_*(x, y, z)] \;\geq\; [\forall u \leq p(x) \exists v \leq q(x) B_*(x, u, v)]$$

is a pair of terms $(a(x, u), b(x, u, z))$ such that [*ignoring the bounds!*]

$$\mathbb{N} \vDash A_*(x, a(x, u), z) \to B_*(x, u, b(x, u, z)).$$

It is $\mathcal{B}$-verifiable iff $\mathcal{B} \vdash A_*(x, a(x, u), z) \to B_*(x, u, b(x, u, z))$.

The Main Theorem (informal)

Let $A(x), B(x)$ be two bounded formulas with some certain complexity, T a bounded theory of arithmetic based on the induction on the bounded formulas in the same complexity class and $\mathcal{B} \subseteq T$ be a base theory. Then TFAE:

- $T \vdash A \Rightarrow B$
- There exists a term $t(x)$ and a sequence of the length $t(x)$ of $\mathcal{B}$-verifiable computational reductions beginning by A and ending in B.

The Main Theorem (informal)

Let $A(x), B(x)$ be two bounded formulas with some certain complexity, T a bounded theory of arithmetic based on the induction on the bounded formulas in the same complexity class and $\mathcal{B} \subseteq T$ be a base theory. Then TFAE:

- $T \vdash A \Rightarrow B$
- There exists a term $t(x)$ and a sequence of the length $t(x)$ of $\mathcal{B}$-verifiable computational reductions beginning by A and ending in B.

Why can it be considered as an algorithm? First observe that if we put $A = \top$ then we have

- $T \vdash B$
- There exists a term $t(x)$ and a sequence of the length $t(x)$ of *$\mathcal{B}$-verifiable* computational reductions beginning by the zero witnessing problem and ending in B.

The Canonical Algorithm

Secondly, note that this sequence of reductions leads to a canonical algorithm:

Example

Ignoring the bounds, let $C_i(x) = \forall y_i \exists z_i D_i(x, y_i, z_i)$ be a sequence of formulas with reductions F_i in between, where D_i's are quantifier-free formulas. Then the sequence of reductions

$$\forall y_0 \exists z_0 D_0(x, y_0, z_0) \geq^{F_0} \ldots \geq^{F_{t(x)-1}} \forall y_{t(x)} \exists z_{t(x)} D_{t(x)}(x, y_{t(x)}, z_{t(x)})$$

leads to the following canonical algorithm: First read $y_{t(x)}$, then use the reduction $F_{t(x)-1}$ to find $y_{t(x)-1}$. Do the same till reaching y_0. Then find z_0 [which is zero for the zero witnessing] and then use the reduction F_0 to find z_1 and do it again till reaching $z_{t(x)}$. This is what we wanted to find.

The Main Theorem (informal)

Let $A(x), B(x)$ be two bounded formulas with some *certain complexity*, T a *bounded theory of arithmetic* based on the induction on the bounded formulas in the same complexity class and $\mathcal{B} \subseteq T$ be a base theory. Then TFAE:

- $T \vdash A \Rightarrow B$
- There exists a term $t(x)$ and a sequence of the length $t(x)$ of $\mathcal{B}$-*verifiable* computational *reductions* beginning by A and ending in B.

To state this theorem more formally, we need the following ingredients:

The Main Theorem (informal)

Let $A(x), B(x)$ be two bounded formulas with some *certain complexity*, T a *bounded theory of arithmetic* based on the induction on the bounded formulas in the same complexity class and $\mathcal{B} \subseteq T$ be a base theory. Then TFAE:

- $T \vdash A \Rightarrow B$
- There exists a term $t(x)$ and a sequence of the length $t(x)$ of $\mathcal{B}$-*verifiable* computational *reductions* beginning by A and ending in B.

To state this theorem more formally, we need the following ingredients:

- The language

The Main Theorem (informal)

Let $A(x), B(x)$ be two bounded formulas with some *certain complexity*, T a *bounded theory of arithmetic* based on the induction on the bounded formulas in the same complexity class and $\mathcal{B} \subseteq T$ be a base theory. Then TFAE:

- $T \vdash A \Rightarrow B$
- There exists a term $t(x)$ and a sequence of the length $t(x)$ of $\mathcal{B}$-*verifiable* computational *reductions* beginning by A and ending in B.

To state this theorem more formally, we need the following ingredients:

- The language
- The hierarchy of bounded formulas

The Main Theorem (informal)

Let $A(x), B(x)$ be two bounded formulas with some *certain complexity*, T a *bounded theory of arithmetic* based on the induction on the bounded formulas in the same complexity class and $\mathcal{B} \subseteq T$ be a base theory. Then TFAE:

- $T \vdash A \Rightarrow B$
- There exists a term $t(x)$ and a sequence of the length $t(x)$ of $\mathcal{B}$-*verifiable* computational *reductions* beginning by A and ending in B.

To state this theorem more formally, we need the following ingredients:

- The language
- The hierarchy of bounded formulas
- The bounded theory of arithmetic

The Main Theorem (informal)

Let $A(x), B(x)$ be two bounded formulas with some *certain complexity*, T a *bounded theory of arithmetic* based on the induction on the bounded formulas in the same complexity class and $\mathcal{B} \subseteq T$ be a base theory. Then TFAE:

- $T \vdash A \Rightarrow B$
- There exists a term $t(x)$ and a sequence of the length $t(x)$ of $\mathcal{B}$*-verifiable* computational *reductions* beginning by A and ending in B.

To state this theorem more formally, we need the following ingredients:

- The language
- The hierarchy of bounded formulas
- The bounded theory of arithmetic
- The reduction

The Main Theorem (informal)

Let $A(x), B(x)$ be two bounded formulas with some *certain complexity*, T a *bounded theory of arithmetic* based on the induction on the bounded formulas in the same complexity class and $\mathcal{B} \subseteq T$ be a base theory. Then TFAE:

- $T \vdash A \Rightarrow B$
- There exists a term $t(x)$ and a sequence of the length $t(x)$ of $\mathcal{B}$-*verifiable* computational *reductions* beginning by A and ending in B.

To state this theorem more formally, we need the following ingredients:

- The language
- The hierarchy of bounded formulas
- The bounded theory of arithmetic
- The reduction
- The flow

The language

Definition *(the language)*

Let $\mathcal{L}$ be a first order language of arithmetic extending

$$\{0, 1, +, -, \cdot, \lfloor \div \rfloor, \leq\}$$

By $\mathcal{R}$ we mean the first order theory consisting of the axioms of commutative discrete ordered semirings (the usual axioms of commutative rings minus the existence of additive inverse plus the axioms to state that $\leq$ is a total discrete order such that $<$ is compatible with addition and multiplication with non-zero elements), plus the following defining axioms for $-$ and $\lfloor \div \rfloor$:

$$(x \geq y \rightarrow (x - y) + y = x) \wedge (x < y \rightarrow x - y = 0)$$

$$((y+1) \cdot \lfloor \frac{x}{y+1} \rfloor \leq x) \wedge (x - (y+1) \cdot \lfloor \frac{x}{y+1} \rfloor < y + 1)$$

Complexity Classes

Definition *(the hierarchy)*

The hierarchy $\{\Sigma_k, \Pi_k\}_{k=0}^{\infty}$ is defined as the following:

(i) $\Pi_0 = \Sigma_0$ is the class of all quantifier-free formulas.

(ii) If $B(x) \in \Sigma_k$ then $\exists x \leq t\ B(x) \in \Sigma_k$ and $\forall x \leq t\ B(x) \in \Pi_{k+1}$.

(iii) If $B(x) \in \Pi_k$ then $\forall x \leq t\ B(x) \in \Pi_k$ and $\forall x \leq t\ B(x) \in \Sigma_{k+1}$.

Complexity Classes

Definition *(the hierarchy)*

The hierarchy $\{\Sigma_k, \Pi_k\}_{k=0}^{\infty}$ is defined as the following:

(i) $\Pi_0 = \Sigma_0$ is the class of all quantifier-free formulas.

(ii) If $B(x) \in \Sigma_k$ then $\exists x \leq t\ B(x) \in \Sigma_k$ and $\forall x \leq t\ B(x) \in \Pi_{k+1}$.

(iii) If $B(x) \in \Pi_k$ then $\forall x \leq t\ B(x) \in \Pi_k$ and $\forall x \leq t\ B(x) \in \Sigma_{k+1}$.

Example

- Define $\mathcal{L}$ as the language consisting of all poly-time functions as function symbols. Then Σ_k characterizes the k-th level of the polytime hierarchy, Σ_k^p.

Complexity Classes

Definition *(the hierarchy)*

The hierarchy $\{\Sigma_k, \Pi_k\}_{k=0}^{\infty}$ is defined as the following:

(i) $\Pi_0 = \Sigma_0$ is the class of all quantifier-free formulas.

(ii) If $B(x) \in \Sigma_k$ then $\exists x \leq t\ B(x) \in \Sigma_k$ and $\forall x \leq t\ B(x) \in \Pi_{k+1}$.

(iii) If $B(x) \in \Pi_k$ then $\forall x \leq t\ B(x) \in \Pi_k$ and $\forall x \leq t\ B(x) \in \Sigma_{k+1}$.

Example

- Define $\mathcal{L}$ as the language consisting of all poly-time functions as function symbols. Then Σ_k characterizes the k-th level of the polytime hierarchy, Σ_k^p.
- As another example, if we add the exponential function to the usual language of arithmetic, then the hierarchy $\{\Sigma_k, \Pi_k\}_{k=0}^{\infty}$ collapses to its very first level $\Sigma_0 = \Pi_0$.

Bounded Arithmetic

Definition *(the theory)*

Let $\mathcal{A} \supseteq \mathcal{R}$ be a set of quantifier-free axioms and $\Phi \in \{\Sigma_k, \Pi_k\}_{k=0}^{\infty}$. By the first order bounded arithmetic, $\mathfrak{B}(\Phi, \mathcal{A})$ we mean the theory in the language $\mathcal{L}$ which consists of axioms $\mathcal{A}$, and the Φ-induction axiom, i.e.

$$A(0) \wedge \forall x(A(x) \to A(x+1)) \to \forall x A(x)$$

where $A \in \Phi$.

Bounded Arithmetic

Definition *(the theory)*

Let $\mathcal{A} \supseteq \mathcal{R}$ be a set of quantifier-free axioms and $\Phi \in \{\Sigma_k, \Pi_k\}_{k=0}^{\infty}$. By the first order bounded arithmetic, $\mathfrak{B}(\Phi, \mathcal{A})$ we mean the theory in the language $\mathcal{L}$ which consists of axioms $\mathcal{A}$, and the Φ-induction axiom, i.e.

$$A(0) \wedge \forall x(A(x) \to A(x+1)) \to \forall x A(x)$$

where $A \in \Phi$.

Example

With our definition of bounded arithmetic, different kinds of theories can be considered as bounded theories of arithmetic, for instance IE_k, IU_k, T_n^k, $I\Delta_0(\mathrm{exp})$ and PRA are just some of the well-known examples.

Computational Reductions

Definition *(the reductions)*

Let $A(\vec{x})$ and $B(\vec{x})$ be some formulas in Π_k and $\{F_i\}_{i=1}^{k}$ be a sequence of terms. By recursion on k, we will define $F = \{F_i\}_{i=1}^{k}$ as a deterministic Π_k-reduction from $B(\vec{x})$ to $A(\vec{x})$ and we will denote it by $A(\vec{x}) \geq^F B(\vec{x})$ when:

(i) If $A(\vec{x}), B(\vec{x})$ are in Π_0, we say that the empty sequence of functions is a reduction from B to A iff $\mathcal{B} \vdash A(\vec{x}) \to B(\vec{x})$.

(ii) If $A = \forall \vec{u} \leq \vec{p}(\vec{x}) C(\vec{x}, \vec{u})$, $B = \forall \vec{v} \leq \vec{q}(\vec{x}) D(\vec{x}, \vec{v})$ and $F = \{F_i\}_{i=1}^{k+1}$ is a sequence of terms, then $A(\vec{x}) \geq^F B(\vec{x})$ iff

$$F_{k+1}(\vec{x}, \vec{v}) \leq \vec{p}(\vec{x}) \to C(\vec{x}, F_{k+1}(\vec{x}, \vec{v})) \geq^{\hat{F}} \vec{v} \leq \vec{q}(\vec{x}) \to D(\vec{x}, \vec{v})$$

where $\hat{F} = \{F_i\}_{i=1}^{k}$.

Computational Reductions

Definition

(*iii*) If $A = \exists \vec{u} \leq \vec{p}(\vec{x}) C(\vec{x}, \vec{u})$, $B = \exists \vec{v} \leq \vec{q}(\vec{x}) D(\vec{x}, \vec{v})$ and $F = \{F_i\}_{i=1}^{k+1}$ is a sequence of terms, then $A(\vec{x}) \geq^F B(\vec{x})$ iff

$$\vec{u} \leq \vec{p}(\vec{x}) \wedge C(\vec{x}, u) \geq^{\hat{F}} F_{k+1}(\vec{x}, \vec{u}) \leq \vec{q}(\vec{x}) \wedge D(\vec{x}, F_{k+1}(\vec{x}, \vec{u}))$$

where $\hat{F} = \{F_i\}_{i=1}^{k}$.

We say B is $(\Pi_k, \mathcal{B})$-reducible to A and we write $A \geq^{(\Pi_k, \mathcal{B})} B$, when there exists a sequence of terms F such that $A \geq^F B$.

Definition *(the flow)*

Let $A(\vec{x}), B(\vec{x}) \in \Pi_k$. A $(\Pi_k, \mathcal{B})$-flow from $A(\vec{x})$ to $B(\vec{x})$ is the following data: A term $t(\vec{x})$, a formula $H(u, \vec{x}) \in \Pi_k$ and sequences of terms E_0, E_1, G_0, G_1 and $F(u)$ such that the following statements are provable in $\mathcal{B}$:

(*i*) $H(0, \vec{x}) \equiv^{(E_0, E_1)} A(\vec{x})$.

(*ii*) $H(t(x), \vec{x}) \equiv^{(G_0, G_1)} B(\vec{x})$.

(*iii*) $\forall u < t(x) H(u, \vec{x}) \geq^{F(u)} H(u+1, \vec{x})$.

If there exists a $(\Pi_k, \mathcal{B})$-flow from $A(\vec{x})$ to $B(\vec{x})$ we will write $A(\vec{x}) \rhd^{(\Pi_k, \mathcal{B})} B(\vec{x})$. Moreover, if Γ and Δ are sequents of formulas in Π_k, by $\Gamma \rhd^{(\Pi_k, \mathcal{B})} \Delta$ we mean $\bigwedge \Gamma \rhd^{(\Pi_k, \mathcal{B})} \bigvee \Delta$.

Definition *(the flow)*

Let $A(\vec{x}), B(\vec{x}) \in \Pi_k$. A $(\Pi_k, \mathcal{B})$-flow from $A(\vec{x})$ to $B(\vec{x})$ is the following data: A term $t(\vec{x})$, a formula $H(u, \vec{x}) \in \Pi_k$ and sequences of terms E_0, E_1, G_0, G_1 and $F(u)$ such that the following statements are provable in $\mathcal{B}$:

(*i*) $H(0, \vec{x}) \equiv^{(E_0, E_1)} A(\vec{x})$.

(*ii*) $H(t(x), \vec{x}) \equiv^{(G_0, G_1)} B(\vec{x})$.

(*iii*) $\forall u < t(x) H(u, \vec{x}) \geq^{F(u)} H(u+1, \vec{x})$.

If there exists a $(\Pi_k, \mathcal{B})$-flow from $A(\vec{x})$ to $B(\vec{x})$ we will write $A(\vec{x}) \rhd^{(\Pi_k, \mathcal{B})} B(\vec{x})$. Moreover, if Γ and Δ are sequents of formulas in Π_k, by $\Gamma \rhd^{(\Pi_k, \mathcal{B})} \Delta$ we mean $\bigwedge \Gamma \rhd^{(\Pi_k, \mathcal{B})} \bigvee \Delta$.

The Main Theorem (A.A.)

Let $\Gamma(\vec{x}) \cup \Delta(\vec{x}) \subseteq \Pi_k$ and $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathfrak{B}(\Pi_k, \mathcal{A})$. Then $\mathfrak{B}(\Pi_k, \mathcal{A}) \vdash \Gamma(\vec{x}) \Rightarrow \Delta(\vec{x})$ iff $\Gamma \rhd^{(\Pi_k, \mathcal{B})} \Delta$.

The Main Lemma

The following lemma establishes a high-level calculus for the flows:

The Main Lemma

- (Conjunction left.) If $\Gamma, A \rhd \Delta$ then $\Gamma, A \wedge B \rhd \Delta$ and $\Gamma, B \wedge A \rhd \Delta$.
- (Conjunction right.) If $\Gamma \rhd \Delta, A$ and $\Gamma \rhd \Delta, B$ then $\Gamma \rhd \Delta, A \wedge B$.
- (Disjunction left.) If $\Gamma, A \rhd \Delta$ and $\Gamma, B \rhd \Delta$ then $\Gamma, A \vee B \rhd \Delta$.
- (Disjunction right.) If $\Gamma \rhd \Delta, A$ then $\Gamma \rhd \Delta, A \vee B$ and $\Gamma \rhd \Delta, B \vee A$.
- (Cut.) If $\Gamma \rhd \Delta, A$ and $\Gamma', A \rhd \Delta'$ then $\Gamma, \Gamma' \rhd \Delta, \Delta'$.
- (Induction.) If $\Gamma, A(x) \rhd \Delta, A(x+1)$ then $\Gamma, A(0) \rhd \Delta, A(t)$.
- (Universal left.) If $\Gamma, A(t) \rhd \Delta$ then $\Gamma, t \leq s, \forall y \leq s\, A(y) \rhd \Delta$.
- (Universal right.) If $\Gamma, y \leq s \rhd \Delta, A(y)$ then $\Gamma \rhd \Delta, \forall y \leq s\, A(y)$.

Main Lemma

- (Existential left.) If $\Gamma, y \leq s, A(y) \rhd \Delta$ then $\Gamma, \exists y \leq s\ A(y) \rhd \Delta$.
- (Existential right.) If $\Gamma \rhd \Delta, A(t)$ then $\Gamma, t \leq s \rhd \Delta, \exists y \leq s\ A(y)$.
- (Contraction left.) If $\Gamma, A, A \rhd \Delta$ then $\Gamma, A \rhd \Delta$.
- (Contraction right.) If $\Gamma \rhd \Delta, A, A$ then $\Gamma \rhd \Delta, A$.

The crucial cases are the quantifier and the contraction rules.

Main Lemma

- (Existential left.) If $\Gamma, y \leq s, A(y) \rhd \Delta$ then $\Gamma, \exists y \leq s\ A(y) \rhd \Delta$.
- (Existential right.) If $\Gamma \rhd \Delta, A(t)$ then $\Gamma, t \leq s \rhd \Delta, \exists y \leq s\ A(y)$.
- (Contraction left.) If $\Gamma, A, A \rhd \Delta$ then $\Gamma, A \rhd \Delta$.
- (Contraction right.) If $\Gamma \rhd \Delta, A, A$ then $\Gamma \rhd \Delta, A$.

The crucial cases are the quantifier and the contraction rules. To have an idea, let us think about the contraction rule:

$$cL\ \frac{\Gamma \rhd \exists y \leq tA(y), \exists y \leq tA(y)}{\Gamma \rhd \exists y \leq tA(y)}$$

To admit this rule it seems reasonable to prove the following reduction

$$[\exists y \leq tA(y) \vee \exists y \leq tA(y)] \geq [\exists y \leq tA(y)]$$

But this means that we have to choose between two witnesses which is not an obvious task without knowing the value of $A(y)$. Note that finding the value of a complex formula $A(y)$ could be costly or even impossible via the terms of the language.

Computability of Characteristic Functions

To solve this problem, we will simulate the decision problem of A by a flow of reductions:

Computability of Characteristic Functions

To solve this problem, we will simulate the decision problem of A by a flow of reductions:

Theorem (Computability of Characteristic Functions)

Let $\{\Sigma_k, \Pi_k\}_{k=0}^{\infty}$ be a hierarchy and $\mathcal{B}$ has characteristic terms for all quantifier-free formulas, then for any $\Psi \in \{\Sigma_k, \Pi_k\}$ if $A(\vec{x}) \in \Psi$ then

$$\rhd^{(\Sigma_{k+1},\mathcal{B})} \exists i \leq 1\, [(i = 1 \to A) \wedge (i = 0 \to \neg A)]$$

Computability of Characteristic Functions

To solve this problem, we will simulate the decision problem of A by a flow of reductions:

Theorem (Computability of Characteristic Functions)

Let $\{\Sigma_k, \Pi_k\}_{k=0}^{\infty}$ be a hierarchy and $\mathcal{B}$ has characteristic terms for all quantifier-free formulas, then for any $\Psi \in \{\Sigma_k, \Pi_k\}$ if $A(\vec{x}) \in \Psi$ then

$$\rhd^{(\Sigma_{k+1}, \mathcal{B})} \exists i \leq 1 \, [(i = 1 \to A) \wedge (i = 0 \to \neg A)]$$

It reduces the problem of deciding A to the problem of deciding the value i which is definitely much easier to handle by terms. The idea behind the theorem is transforming the usual brute-force algorithm [open all quantifiers and check all the possibilities] to a sequence of simple term-based reductions.

The proof of the Main Theorem

The Main Theorem (A.A.)

Let $\Gamma(\vec{x}) \cup \Delta(\vec{x}) \subseteq \Pi_k$ and $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathfrak{B}(\Pi_k, \mathcal{A})$. Then TFAE:

- $\mathfrak{B}(\Pi_k, \mathcal{A}) \vdash \Gamma(\vec{x}) \Rightarrow \Delta(\vec{x})$
- $\Gamma \rhd^{(\Pi_k, \mathcal{B})} \Delta$.

Proof Sketch of the Main Theorem

If $\Gamma \rhd^{(\Pi_k, \mathcal{B})} \Delta$ then there exists a formula $H(u, \vec{x}) \in \Pi_k$ such that $H(i+1, \vec{x})$ is reducible to $H(i, \vec{x})$ provable in $\mathcal{B}$. Hence $\mathcal{B} \vdash H(i, \vec{x}) \Rightarrow H(i+1, \vec{x})$. Since $\mathcal{B} \subseteq \mathfrak{B}(\Pi_k, \mathcal{A})$ and we have Π_k-induction in $\mathfrak{B}(\Pi_k, \mathcal{A})$ we have $\mathfrak{B}(\Pi_k, \mathcal{A}) \vdash H(0, \vec{x}) \Rightarrow H(t(\vec{x}), \vec{x})$ hence $\mathfrak{B}(\Pi_k, \mathcal{A}) \vdash A \Rightarrow B$.

The proof of the Main Theorem

The Main Theorem (A.A.)

Let $\Gamma(\vec{x}) \cup \Delta(\vec{x}) \subseteq \Pi_k$ and $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathfrak{B}(\Pi_k, \mathcal{A})$. Then TFAE:

- $\mathfrak{B}(\Pi_k, \mathcal{A}) \vdash \Gamma(\vec{x}) \Rightarrow \Delta(\vec{x})$
- $\Gamma \rhd^{(\Pi_k, \mathcal{B})} \Delta$.

Proof Sketch of the Main Theorem

If $\Gamma \rhd^{(\Pi_k, \mathcal{B})} \Delta$ then there exists a formula $H(u, \vec{x}) \in \Pi_k$ such that $H(i+1, \vec{x})$ is reducible to $H(i, \vec{x})$ provable in $\mathcal{B}$. Hence $\mathcal{B} \vdash H(i, \vec{x}) \Rightarrow H(i+1, \vec{x})$. Since $\mathcal{B} \subseteq \mathfrak{B}(\Pi_k, \mathcal{A})$ and we have Π_k-induction in $\mathfrak{B}(\Pi_k, \mathcal{A})$ we have $\mathfrak{B}(\Pi_k, \mathcal{A}) \vdash H(0, \vec{x}) \Rightarrow H(t(\vec{x}), \vec{x})$ hence $\mathfrak{B}(\Pi_k, \mathcal{A}) \vdash A \Rightarrow B$. For the other direction, we have to show that the flow interpretation admits all the rules in the sequent calculus of the bounded theory of arithmetic which is nothing but the content of the main lemma.

Applications I

For the first application, consider the hierarchy $\{IU_k\}_{k=0}^{\infty}$ written in the language augmented with subtraction and division. These are the fragments of the theory $I\Delta_0$ that are related to the computational world of linear time hierarchy. Moreover, consider the class of all functions constructed from zero, projections and closed under successor, addition, production, subtraction and division and call it R:

Theorem

Let $\Gamma(\vec{x}) \cup \Delta(\vec{x}) \subseteq U_k$, then $IU_k \vdash \Gamma(\vec{x}) \Rightarrow \Delta(\vec{x})$ iff $\Gamma \rhd^{(U_k, \mathcal{R})} \Delta$.

The second condition means that there exists a sequence of length $t \in R$ of formulas in U_k beginning from $\bigwedge \Gamma$ ending with $\bigvee \Delta$ such that each formula is reducible to its successor provably in $\mathcal{R}$ and using just the functions in R.

Applications II

Theorem

Let $\Gamma(\vec{x}) \cup \Delta(\vec{x}) \subseteq \hat{\Pi}^b_k(\#_n)$, *then* $T^k_n \vdash \Gamma(\vec{x}) \Rightarrow \Delta(\vec{x})$ *iff* $\Gamma \rhd^{(\hat{\Pi}^b_k(\#_n), \mathrm{PV}(\#_n))} \Delta$. *Specifically, for* $n = 2$, $T^k_2 \vdash \Gamma(\vec{x}) \Rightarrow \Delta(\vec{x})$ *iff* $\Gamma \rhd^{(\hat{\Pi}^b_k, \mathrm{PV})} \Delta$.

The second condition in the latter case means that there exists a sequence of length $2^{p(|\vec{x}|)}$ of formulas in Π^b_k beginning from $\bigwedge \Gamma$ ending in $\bigvee \Delta$ such that each formula is reducible to its successor provably in PV and using just the polynomial time computable functions.

Applications II

Theorem

Let $\Gamma(\vec{x}) \cup \Delta(\vec{x}) \subseteq \hat{\Pi}^b_k(\#_n)$*, then* $T^k_n \vdash \Gamma(\vec{x}) \Rightarrow \Delta(\vec{x})$ *iff* $\Gamma \rhd^{(\hat{\Pi}^b_k(\#_n), \mathrm{PV}(\#_n))} \Delta$*. Specifically, for* $n = 2$*,* $T^k_2 \vdash \Gamma(\vec{x}) \Rightarrow \Delta(\vec{x})$ *iff* $\Gamma \rhd^{(\hat{\Pi}^b_k, \mathrm{PV})} \Delta$*.*

The second condition in the latter case means that there exists a sequence of length $2^{p(|\vec{x}|)}$ of formulas in Π^b_k beginning from $\bigwedge \Gamma$ ending in $\bigvee \Delta$ such that each formula is reducible to its successor provably in PV and using just the polynomial time computable functions. This theorem provides another characterization for all total NP search problems of T^k_2 as well as all higher order search problems in these theories. In fact, the very notion of a flow as we explained here is a generalization of higher PLS problems of Buss and Beckmann on the one hand and Skelly and Thapen's game induction principles, on the other. Both of these problems have been successfully used to characterize the search problems of the hierarchy T^k_2.

Thank you for your attention!