Non-deterministic Flows in Arithmetic

Amirhossein Akbar Tabatabai
Utrecht University, the Netherlands

Abstract

In [1], we have developed a theory of deterministic flows to extend the spirit of the Dialectica interpretation to weak bounded theories of arithmetic. In this paper, we will generalize that theory to introduce the notion of a non-deterministic flow as a sequence of computational problems together with a sequence of non-deterministic version of game reductions. It provides a proof mining method for a general notion of bounded arithmetic which turns out helpful in program extractions. More specifically, it leads to a characterization of all the higher total search problems in all the higher order bounded theories of arithmetic including the interesting case of $\forall\hat{\Sigma}^b_j$-consequences of the hierarchy S^k_2, for $1 \leq j \leq k$. Moreover, the method is also useful to provide another proof for the strong witnessing theorem for the theories S^k_2.

1 Introduction

One of the successful trends in proof theory is and always has been the search for the more and more strong relationships between proofs and computations. These relationships are witnessed by different extracting techniques that bring out informative algorithms from given proofs in given theories. These methods cover a wide spectrum, from witnessing techniques in bounded theories of arithmetic to ordinal analysis and functional interpretation of weak set theoretical systems.

In this paper, we will develop one of these extracting methods, designed specifically for bounded theories of arithmetic. This method is based on two simple ingredients: First reducing any proof in any bounded theory of arithmetic to a single uniform sequence of implications such that these implications become provable in a very weak theory (usually universal induction-free

system powerful enough to handle the four basic mathematical operations, addition, subtraction, multiplication and division). And secondly, developing a program interpretation of the Herbrand theorem adopted for our bounded arithmetical language to witness any step in the implications by the terms of the language. These programs that we call *reduction programs* generalize the usual reduction between k-turn games and teacher-student interactive protocols. In fact, they provide a non-deterministic interactive machinery to witness the existential variables by the universal variables using the terms of the language.

As a result of combining these ingredients, we will establish a general method to extract computational information from bounded arithmetical proofs. As an application of this method, we will first propose a characterization of all total search problems of any complexity in any bounded arithmetical theory, especially in the presence of higher smash functions. These theories mimics the higher order bounded theories in a first order setting and hence our characterization can be interpreted as a characterization of all provably total higher search problems in higher order bounded arithmetic. More specifically, we will investigate the bounded statements of the theories S_2^k for $k \geq 1$ to reduce their provability to a polynomially long sequence of reduction programs between k-turn games. We will also apply our technique to reprove the strong witnessing theorem for theories S_2^k. This type of witnessing theorem has been investigated for different bounded theories (See [8], [3], [2], [5] and [9]). Here, we propose another proof for this result for the hierarchy $\{S_2^k\}_{k\geq 1}$.

2 Preliminaries

In this section we will review some preliminaries. First, let us fix a language which can be any arbitrary extension of a ring-type language for numbers:

Definition 2.1. Let $\mathcal{L}$ be a first order language of arithmetic extending $\mathcal{L}_{\mathcal{R}} = \{0, 1, +, \dot{-}, \cdot, d(-,-), \leq\}$ where $x \dot{-} y$ and $d(x, y)$ mean $max\{0, x - y\}$ and $\lfloor \frac{x}{y+1} \rfloor$ in the standard model, respectively. By $\mathcal{R}$ we mean the first order theory in the language $\mathcal{L}_{\mathcal{R}}$ 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 $\dot{-}$ and d:

$$(x \geq y \to (x \dot{-} y) + y = x) \wedge (x < y \to x \dot{-} y = 0),$$

and

$$((y+1) \cdot d(x,y) \leq x) \wedge (x \dot{-} (y+1) \cdot d(x,y) < y+1).$$

Note that to avoid division by zero and to have a total function symbol in the language we defined division as $\lfloor \frac{x}{y+1} \rfloor$ and not $\lfloor \frac{x}{y} \rfloor$.

Remark 2.2. First note that $\mathcal{R}$ can prove that all elements are non-negative simply because multiplying them preserves the order. Secondly note that the language $\mathcal{L}$ is powerful enough to represent the conditional function

$$C(x,y,z) = \begin{cases} y & x = 0 \\ z & x > 0 \end{cases}$$

as a term and $\mathcal{R}$ is powerful enough to prove that the term works. The crucial point is that the term $\chi_{=0}(x) = d(x+2,x) \dot{-} 1 = \lfloor \frac{x+2}{x+1} \rfloor \dot{-} 1$ has the following property provably in $\mathcal{R}$:

$$\chi_{=0}(x) = \begin{cases} 1 & x = 0 \\ 0 & x > 0 \end{cases}$$

Hence it is enough to represent C by $\chi(x)y + (1 \dot{-} \chi(x))z$. Moreover, using $\chi_{\leq}(x,y) = \chi_{=0}(x \dot{-} y)$, we can represent the characteristic function for $\leq$ and since we have the power to simulate all boolean operators and $x = y$ is equivalent to $x \leq y \wedge y \leq x$, we have the characteristic functions of all quantifier-free formulas of the language $\mathcal{L}_{\mathcal{R}} = \{0, 1, +, \dot{-}, \cdot, d(-,-), \leq\}$.

To define different bounded systems of arithmetic, we have to set two main ingredients of the induction axiom, i.e., the complexity of the induction formula and the length of the induction. For the first one we have:

Definition 2.3. (i) A class of bounded formulas Π is called a π-class of the language $\mathcal{L}$ if it includes all quantifier-free formulas of $\mathcal{L}$, is closed under substitutions, subformulas, conjunction, disjunction and bounded universal quantifiers and if $\exists y \leq t\ B(y) \in \Pi$ then there exists $C(y)$ such that $\vdash C(y) \leftrightarrow \neg B(y)$ and $\forall y \leq t\ C(y) \in \Pi$.

(ii) A class of bounded formulas Σ is called a σ-class of the language $\mathcal{L}$ if it includes all quantifier-free formulas of $\mathcal{L}$, is closed under substitutions, subformulas, conjunction, disjunction and bounded existential quantifiers and if $\forall y \leq t\ B(y) \in \Sigma$ then there exists $C(y)$ such that $\vdash C(y) \leftrightarrow \neg B(y)$ and $\exists y \leq t\ C(y) \in \Sigma$.

Example 2.4. The class of all bounded formulas is a trivial example of both π and σ classes. The more interesting examples though include the classes $\hat{\Pi}^b_k(\#_m)$ and $\hat{\Sigma}^b_k(\#_m)$ (drop $\#_m$ from the notation when $m = 2$), in the language of bounded arithmetic augmented with subtraction, division and $\#_i$ for $2 \leq i \leq m$. These classes are defined in the following way:

(*i*) $\hat{\Pi}^b_0(\#_m) = \hat{\Sigma}^b_0(\#_m)$ is the class of all sharply bounded formulas, i.e., the formulas whose quantifiers are bounded by a term of the form $|t|$, for some term t,

(*ii*) $\hat{\Sigma}^b_k(\#_m) \subseteq \hat{\Sigma}^b_{k+1}(\#_m)$ and $\hat{\Pi}^b_k(\#_m) \subseteq \hat{\Pi}^b_{k+1}(\#_m)$,

(*iii*) $\hat{\Pi}^b_k(\#_m)$ and $\hat{\Sigma}^b_k(\#_m)$ are closed under conjunction and disjunction,

(*iv*) If $B(x) \in \hat{\Sigma}^b_k(\#_m)$ then $\exists x \leq t\ B(x) \in \hat{\Sigma}^b_k(\#_m)$ and $\forall x \leq t\ B(x) \in \hat{\Pi}^b_{k+1}(\#_m)$ and

(*v*) If $B(x) \in \hat{\Pi}^b_k(\#_m)$ then $\forall x \leq t\ B(x) \in \hat{\Pi}^b_k(\#_m)$ and $\exists x \leq t\ B(x) \in \hat{\Sigma}^b_{k+1}(\#_m)$.

We can also consider a more relaxed version of these classes, i.e., $\Sigma^b_k(\#_m)$ and $\Pi^b_k(\#_m)$, (again dropping $\#_m$ when $m = 2$), which are defined with the same definition as above, adding the condition that:

"$\Pi^b_k(\#_m)$ and $\Sigma^b_k(\#_m)$ are closed under sharply bounded quantification, i.e., a quantification bounded by $|t|$ for some term t."

Note that, assuming that the polynomial hierarchy does not collapse, these more relaxed versions of the classes (for $m = 2$) are not π- and σ-classes, respectively. The reason is the existence of a Π^b_k formula (a Σ^b_k formula), ending with an existential (a universal) sharply bounded quantifier, which is also bounded, without a Π^b_k negation (a Σ^b_k negation).

Now let us define a robust form for the classes of terms that can play the role of induction-length.

Definition 2.5. Let $\mathcal{A} \supseteq \mathcal{R}$ be a theory. A class of terms, $\mathbb{T}$, is called an $\mathcal{A}$-term ideal if:

(*i*) It is closed under all function symbols of the language $\mathcal{L}_\mathcal{R}$, provably in $\mathcal{A}$, i.e. for any function symbol $f \in \mathcal{L}_\mathcal{R}$ and any $t(\vec{x}) \in \mathbb{T}$, there exist $r(\vec{x}) \in \mathbb{T}$ such that $\mathcal{A} \vdash r(\vec{x}) = f(t(\vec{x}))$.

(*ii*) It is closed under substitution, i.e. if $t(\vec{x}, y) \in \mathbb{T}$ and s is an arbitrary term (not necessarily in $\mathbb{T}$) then $t(\vec{x}, s) \in \mathbb{T}$ provably in $\mathcal{A}$, i.e. there exists $r(\vec{x}) \in \mathbb{T}$ such that $\mathcal{A} \vdash r(\vec{x}) = t(\vec{x}, s)$.

(*iii*) It has a subset of monotone majorizing terms provably in $\mathcal{A}$, i.e. there exists a set of terms $M \subseteq \mathbb{T}$ such that for any $t(\vec{x}) \in \mathbb{T}$ there exists $s(\vec{x}) \in M$ such that $\mathcal{A} \vdash t(\vec{x}) \leq s(\vec{x})$ and for any $r(\vec{x}) \in M$, $\mathcal{A} \vdash \vec{x} \leq \vec{y} \rightarrow r(\vec{x}) \leq r(\vec{y})$.

Example 2.6. For the language $\mathcal{L}_{\mathcal{R}}$, there are two trivial $\mathcal{R}$-term ideals; $\mathbb{T}_{all}$ consisting of all terms of the language and $\mathbb{T}_{cl}$ consisting of all closed terms, with majorizing sets as the set of all polynomials and the whole set of closed terms, respectively. To have a non-trivial example, consider the language of bounded arithmetic extended with subtraction and division and the theory $\mathcal{A}_p$ as BASIC + $\mathcal{R}$ plus the axioms $|x| \leq x$, $|xy| \leq |x| + |y|$ and $x \leq y \rightarrow |x| \leq |y|$. Now define $\mathbb{T}_p$ as the class of all terms majorized by a term in the form $p(|\vec{x}|)$ for some polynomial p provably in $\mathcal{A}_p$. The majorizing subset is the set of all terms in the form $p(|\vec{x}|)$ and the reason that the set is an $\mathcal{A}_p$-ideal is that all terms are bounded by a polynomial in length and the fact that these terms are increasing, both provably in $\mathcal{A}_p$.

Using these ingredients, we can introduce the general definition of a bounded theory of arithmetic:

Definition 2.7. Let $\mathcal{A} \supseteq \mathcal{R}$ be a set of quantifier-free axioms, $\mathbb{T}$ be an $\mathcal{A}$-term ideal and Φ be a class of bounded formulas closed under substitution and subformulas. By the first order bounded arithmetic, $\mathfrak{B}(\mathbb{T}, \Phi, \mathcal{A})$ we mean the theory in the language $\mathcal{L}$ which consists of axioms $\mathcal{A}$, and the $(\mathbb{T}, \Phi)$-induction axiom, i.e.,

$$A(0) \wedge \forall x(A(x) \rightarrow A(x+1)) \rightarrow \forall x A(t(x)),$$

where $A \in \Phi$ and $t \in \mathbb{T}$.

Example 2.8. With our definition of bounded arithmetic, different kinds of theories can be considered as bounded theories of arithmetic, for instance $I\Delta_0$, S^k_n, T^k_n, $I\Delta_0(\exp)$ and PRA augmented with subtraction and division in the language and the axioms of $\mathcal{R}$ in the theory, are just some of the well-known examples.

Remark 2.9. Note that the theory $\mathfrak{B}(\mathbb{T}, \Phi, \mathcal{A})$ may not have access to the full-induction scheme

$$A(0) \wedge \forall x(A(x) \rightarrow A(x+1)) \rightarrow \forall x A(x),$$

for any $A \in \Phi$. For instance, in the theory S_2^k, the system only has the length-induction that is believed to be weaker than the usual induction in T_2^k.

As usual in the proof theoretical investigations, we are interested in a more structural representation of proofs. For this purpose and for any arbitrary set $\mathbf{Ax}$ of sequents, consider the system $G1(\mathbf{Ax})$ consisting of the following rules:

Axioms:

$$\frac{}{A \Rightarrow A} \qquad \frac{}{A_1, \ldots A_n \Rightarrow B_1, \ldots B_m}$$

where the right axiom is a substitution of a sequent in $\mathbf{Ax}$.

Structural Rules:

$$(wL)\ \frac{\Gamma \Rightarrow \Delta}{\Gamma, A \Rightarrow \Delta} \qquad (wR)\ \frac{\Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, A}$$

$$(cL)\ \frac{\Gamma, A, A \Rightarrow \Delta}{\Gamma, A \Rightarrow \Delta} \qquad (cR)\ \frac{\Gamma \Rightarrow \Delta, A, A}{\Gamma \Rightarrow \Delta, A}$$

$$(cut)\ \frac{\Gamma_0 \Rightarrow \Delta_0, A \qquad \Gamma_1, A \Rightarrow \Delta_1}{\Gamma_0, \Gamma_1 \Rightarrow \Delta_0, \Delta_1}$$

Propositional Rules:

$$\vee L\ \frac{\Gamma_0, A \Rightarrow \Delta_0 \qquad \Gamma_1, B \Rightarrow \Delta_1}{\Gamma_0, \Gamma_1, A \vee B \Rightarrow \Delta_0, \Delta_1} \qquad \vee R\ \frac{\Gamma \Rightarrow \Delta, A_i}{\Gamma \Rightarrow \Delta, A_0 \vee A_1}\ (i = 0, 1)$$

$$\wedge L\ \frac{\Gamma, A_i \Rightarrow \Delta}{\Gamma, A_0 \wedge A_1 \Rightarrow \Delta}\ (i = 0, 1) \qquad \wedge R\ \frac{\Gamma_0 \Rightarrow \Delta_0, A \qquad \Gamma_1 \Rightarrow \Delta_1, B}{\Gamma_0, \Gamma_1 \Rightarrow \Delta_0, \Delta_1, A \wedge B}$$

$$\rightarrow L\ \frac{\Gamma_0 \Rightarrow A, \Delta_0 \qquad \Gamma_1, B \Rightarrow \Delta_1}{\Gamma_0, \Gamma_1, A \rightarrow B \Rightarrow \Delta_0, \Delta_1} \qquad \rightarrow R\ \frac{\Gamma, A \Rightarrow B, \Delta}{\Gamma \Rightarrow \Delta, A \rightarrow B}$$

$$\neg L\ \frac{\Gamma \Rightarrow \Delta, A}{\Gamma, \neg A \Rightarrow \Delta} \qquad \neg R\ \frac{\Gamma, A \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \neg A}$$

Quantifier rules:

$$\forall L\ \frac{\Gamma, A(s) \Rightarrow \Delta}{\Gamma, \forall y\ A(y) \Rightarrow \Delta} \qquad \forall R\ \frac{\Gamma \Rightarrow \Delta, A(b)}{\Gamma \Rightarrow \Delta, \forall y\ A(y)}$$

$$\exists L\ \frac{\Gamma, A(b) \Rightarrow \Delta}{\Gamma, \exists y\ A(y) \Rightarrow \Delta} \qquad \exists R\ \frac{\Gamma \Rightarrow \Delta, A(s)}{\Gamma, \Rightarrow \Delta, \exists y\ A(y)}$$

Bounded Quantifier rules:

$$\forall^{\leq} L\ \frac{\Gamma, A(s) \Rightarrow \Delta}{\Gamma, s \leq t, \forall y \leq t\ A(y) \Rightarrow \Delta} \qquad \forall^{\leq} R\ \frac{\Gamma, b \leq t \Rightarrow \Delta, A(b)}{\Gamma \Rightarrow \Delta, \forall y \leq t\ A(y)}$$

$$\exists^{\leq} L\ \frac{\Gamma, b \leq t, A(b) \Rightarrow \Delta}{\Gamma, \exists y \leq t\ A(y) \Rightarrow \Delta} \qquad \exists^{\leq} R\ \frac{\Gamma \Rightarrow \Delta, A(s)}{\Gamma, s \leq t, \Rightarrow \Delta, \exists y \leq t\ A(y)}$$

Note that in the rules $(\forall R)$, $(\exists L)$, $(\forall^{\leq} R)$ and $(\exists^{\leq} L)$, the variable b must not occur in the lower sequent of the rule.

There is also another type of sequent calculus, called $G3(\mathbf{Ax})$, absorbing all the structural rules. It is defined with the same rules, by eliminating structural rules and replacing the axioms, the cut rule, the propositional rules and the rules $(\forall L)$, $(\exists R)$, $(\forall^{\leq} L)$ and $(\exists^{\leq} R)$ by the following rules:

Axioms:

$$\frac{}{\Gamma, P \Rightarrow P, \Delta} \qquad \frac{}{\Gamma, P_1, \ldots P_n \Rightarrow Q_1, \ldots Q_m, \Delta}$$

where $P_1, \ldots P_n \Rightarrow Q_1, \ldots Q_m \in cl(\mathbf{Ax})$ and P, P_i's and Q_j's are all atomic formulas. By $cl(\mathbf{Ax})$ we mean the closure of $\mathbf{Ax}$ under substitution and contraction.

Structural Rules:

$$(cut)\ \frac{\Gamma \Rightarrow \Delta, A \qquad \Gamma, A \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}$$

where P is an atmoic formula, and

Propositional Rules:

$$\vee L\ \frac{\Gamma, A \Rightarrow \Delta \qquad \Gamma, B \Rightarrow \Delta}{\Gamma, A \vee B \Rightarrow \Delta} \qquad \vee R\ \frac{\Gamma \Rightarrow \Delta, A, B}{\Gamma \Rightarrow \Delta, A \vee B}$$

$$\wedge L\ \frac{\Gamma, A, B \Rightarrow \Delta}{\Gamma, A \wedge B \Rightarrow \Delta, C} \qquad \wedge R\ \frac{\Gamma \Rightarrow \Delta, A \qquad \Gamma \Rightarrow \Delta, B}{\Gamma \Rightarrow \Delta, A \wedge B}$$

$$\rightarrow L\ \frac{\Gamma \Rightarrow A, \Delta \qquad \Gamma, B \Rightarrow \Delta}{\Gamma, A \rightarrow B \Rightarrow \Delta} \qquad \rightarrow R\ \frac{\Gamma, A \Rightarrow B, \Delta}{\Gamma \Rightarrow \Delta, A \rightarrow B}$$

Quantifier rules:

$$\forall L\ \frac{\Gamma, A(s), \forall y\ A(y) \Rightarrow \Delta}{\Gamma, \forall y\ A(y) \Rightarrow \Delta} \qquad \exists R\ \frac{\Gamma \Rightarrow \Delta, A(s), \exists y\ A(y)}{\Gamma, \Rightarrow \Delta, \exists y\ A(y)}$$

Bounded Quantifier rules:

$$\forall^{\leq} L\ \frac{\Gamma, A(s), \forall y \leq t\ A(y) \Rightarrow \Delta}{\Gamma, s \leq t, \forall y \leq t\ A(y) \Rightarrow \Delta} \qquad \exists^{\leq} R\ \frac{\Gamma \Rightarrow \Delta, A(s), \exists y \leq t\ A(y)}{\Gamma, s \leq t, \Rightarrow \Delta, \exists y \leq t\ A(y)}$$

Using the system $G1$, choosing $\mathbf{Ax}$ as the set of all sequents $(\Rightarrow A)$ where $A \in \mathcal{A}$ and adding the following induction rule to $G1(\mathbf{Ax})$:

Induction:

$$(Ind)\ \frac{\Gamma, A(b) \Rightarrow \Delta, A(b+1)}{\Gamma, A(0) \Rightarrow \Delta, A(t)}$$

for every $A \in \Phi$ and $t \in \mathbb{T}$, we can capture the theory $\mathfrak{B}(\mathbb{T}, \Phi, \mathcal{A})$. Note that in the induction rule the variable b must not occur in the lower sequent of the rule.

The most important property of the sequent calculi that we have defined is cut elimination:

Theorem 2.10. *(Cut Elimination)*

(*i*) *Any proof in the systems* $G1(\boldsymbol{Ax})$ *and* $G3(\boldsymbol{Ax})$ *can be transformed to a proof in which every cut rule has at least one premise chosen from the axioms of* $\boldsymbol{Ax}$.

(*ii*) *If* $\mathfrak{B}(\mathbb{T}, \Phi, \mathcal{A}) \vdash \Gamma \Rightarrow \Delta$ *then there exists a free-cut free proof for the same sequent in the same system.*

Corollary 2.11. *If* $\Gamma \cup \Delta \subseteq \Phi$ *and* $\mathfrak{B}(\mathbb{T}, \Phi, \mathcal{A}) \vdash \Gamma \Rightarrow \Delta$ *then there exists a proof of the same sequent in the same system such that all formulas occurring in the proof are in* Φ.

The proofs of the Theorem 2.10 and the Corollary 2.11 can be essentially found in [7] and [4].

3 Non-deterministic Reductions and Reduction Programs

Let us begin right away by non-deterministic reductions.

Definition 3.1. Let $\mathcal{B}$ be a theory and $A(\vec{x})$ and $B(\vec{x})$ be some formulas in the language $\mathcal{L}$. We say $B(\vec{x})$ is non-deterministically $\mathcal{B}$-reducible to $A(\vec{x})$ and we write $A(\vec{x}) \geq_n^{\mathcal{B}} B(\vec{x})$ if $\mathcal{B} \vdash A(\vec{x}) \to B(\vec{x})$. Moreover, by the equivalence $A \equiv_n^{\mathcal{B}} B$ we mean the conjunction of $A \geq_n^{\mathcal{B}} B$ and $B \geq_n^{\mathcal{B}} A$.

The natural question is that how this proof-theoretic concept can be considered as a computational reduction and why it is called non-deterministic. To answer this question, first recall that by the flow machinery, we intend to transform any arithmetical proof to a sequence of reductions, and the base theory for those reductions preferably is a simple universal and possibly induction-free theory. Therefore, we can use the Herbrand theorem for each step of the reduction to witness the essentially existential quantifiers in $A \to B$ by its universal quantifiers. This is actually what is happening in the deterministic reductions, but here the difference is the use of $\vee$-expansions in the Herbrand proof. Intuitively, these expansions allow us to use some constant many terms to witness one existential quantifier as opposed to just one term in the case of deterministic reductions. Moreover, expansions make some room for interaction in providing the witnessing terms which makes the concrete witnesses extremely complicated. For these reasons, we call these reductions non-deterministic and in the following we try to state a computational interpretation of Herbrand theorem, tailored specifically for our setting here.

Definition 3.2. Let $\mathcal{L}$ be a language extending the language of $\mathcal{L}_{\mathcal{R}}$. A formula $A(\vec{x})$ is in the *prenex bounded form* if there exists a quantifier-free formula G_A, such that

$$A = \forall y_1 \leq p_1(\vec{x}) \exists z_1 \leq q_1(\vec{x}) \forall y_2 \leq p_2(\vec{x}) \dots G_A(\vec{x}, y_1, z_1, y_2, z_2, \dots)$$

Note that all bounding terms depend only on $\vec{x}$ where $\vec{x}$ is the set of all free variables of $A(\vec{x})$. Moreover, we say that the formula is in the *k-prenex bounded form* when the number of bounded quantifiers is at most k.

Definition 3.3. Let $\mathcal{L}$ be a language extending the language of $\mathcal{L}_{\mathcal{R}}$, the formulas

$$\{\forall y_{i1} \leq p_{i1}(\vec{x}) \exists z_{i1} \leq q_{i1}(\vec{x}) \forall y_{i2} \leq p_{i2}(\vec{x}) \dots G_i(\vec{x}, y_{i1}, z_{i1}, y_{i2}, z_{i2}, \dots)\}_{i \in I}$$

and

$$\{\forall u_{j1} \leq p'_{j1}(\vec{x})\exists v_{j1} \leq q'_{j1}(\vec{x})\forall u_{j2} \leq p'_{j2}(\vec{x}) \dots H_j(\vec{x}, u_{j1}, v_{j1}, u_{j2}, v_{j2}, \dots)\}_{j\in J}$$

be in the prenex bounded form where $\vec{x}$, y_{in}, z_{in}, u_{jm} and v_{jm} are distinct variables and $\mathcal{B}$ be a theory extending $\mathcal{R}$. Define $\mathcal{V}$ as the set of distinct variables y^k_{in}, z^k_{in}, u^k_{jm} and v^k_{jm} for $k \geq 0$. These variables provide infinite many copies of any variable from y_{in}, z_{in}, u_{jm} and v_{jm}. Moreover, note that for $k = 0$ we have the original copy, i.e., $y^0_{in} = y_{in}$, $z^0_{in} = z_{in}$, $u^0_{jm} = u_{jm}$ and $v^0_{jm} = v_{jm}$.

Consider the instructions [**Read** $X \leq t(\vec{x})$] and [**Compute** Y **by** $s \leq t(\vec{x})$] where t is a term depending only on $\vec{x}$ and variables X and Y are chosen from the set $\mathcal{V}$. By a *$\mathcal{B}$-reduction program* from $\{H_j\}_{j\in J}$ to $\{G_i\}_{i\in I}$ we mean a sequence $P = (P_r)^l_{r=0}$ of instructions such that:

(i) The instruction [**Read** $X \leq t(\vec{x})$] applies only on $X = u^k_{jm}$ and $X = z^k_{in}$ variables.

(ii) The instruction [**Compute** Y **by** $s \leq t(\vec{x})$] applies only on $Y = v^k_{jm}$ and $Y = y^k_{in}$ variables.

(iii) Any variable can be read or computed at most once.

(iv) We can read or compute a variable Z if there exists a decreasing path of already read or computed variables starting from Z and ending in one of the variables $\{y_{i1}\}_{i\in I}$ or $\{u_{j1}\}_{j\in J}$. By "decreasing", we refer to the order defined by the relations $(y^k_{in} \prec z^k_{in})$, $(z^k_{in} \prec y^k_{i(n+1)})$, $(u^k_{jm} \prec v^k_{jm})$, $(v^k_{jm} \prec u^k_{j(m+1)})$ and $(Y^k \prec Y^{k+1})$ for any $Y \in \{v_{jm}, y_{in}\}$.

(v) In the instruction [**Compute** Y **by** $s \leq t(\vec{x})$], the term s depends only on the variables $\vec{x}$ and the variables that had been read before the current stage. Moreover, we have to have $\mathcal{B} \vdash \forall \vec{X} \leq \vec{r}(\vec{x})s(\vec{X}) \leq t(\vec{x})$ where $\vec{X}$ are the previously read variables and $\vec{r}(\vec{x})$ are their corresponding bounds.

(vi) For the last condition, first define $S(P^{<r})$ recursively as:

$$S(P^{<0}) = \{G_i\}_{i\in I} \Rightarrow \{H_j\}_{j\in J}$$

and $S(P^{<r+1})$ is defined from $S(P^{<r})$ by the following rule:

There are two cases to consider. First if P_r is the instruction [**Read** $X \leq t(\vec{x})$], then replace all instances of $\forall X \leq t(\vec{x})A(X)$ in the succedent of $S(P^{<r})$ by $A(X)$. And also replace all instances of $\exists X \leq t(\vec{x})A(X)$ in the precedent of $S(P^{<r})$ again by $A(X)$. Second, if P_r is the instruction [**Compute** Y **by** $s \leq t(\vec{x})$], then for any occurrence of $\exists Y \leq t(\vec{x})A(Y)$ in the succedent of $S(P^{<r})$, replace $\exists Y \leq t(\vec{x})A(Y)$ by $(\exists Y \leq t(\vec{x})A(Y))^{+1}$ and add $A(s)$ to the succedent of $S(P^{<r})$. And for any occurrence of $\forall Y \leq t(\vec{x})A(Y)$ in the precedent, replace $\forall Y \leq t(\vec{x})A(Y)$ by $(\forall Y \leq t(\vec{x})A(Y))^{+1}$ and add $A(s)$ in the precedent, where C^{+1} means increasing the upper index of any bounded variable in C by one.

Now after defining $S(P^{<r})$, we also have to have the following last condition: There should be a quantifier-free sub-sequent $S' = (\Gamma \Rightarrow \Delta)$ of $S(P^{<l+1})$ such that $\forall \vec{X} \leq \vec{r}(\vec{x})(\bigwedge \Gamma \rightarrow \bigvee \Delta)$ is provable in $\mathcal{B}$, where $\vec{X}$ are all the read variables occurred in S' and $\vec{r}(\vec{x})$ are their corresponding bounds.

Remark 3.4. *(Game Interpretation)* Let

$$C = \forall y_1 \leq p_1(\vec{x}) \exists z_1 \leq q_1(\vec{x}) \forall y_2 \leq p_2(\vec{x}) \dots G_C(\vec{x}, y_1, z_1, y_2, z_2, \dots)$$

be in the k-prenex bounded form with exactly k quantifiers. The game associated to this formula, $\mathcal{G}_C$, is defined as the following: There are two players. The first player chooses a number $y_1 \leq p_1(\vec{x})$, then the second player chooses a number $z_1 \leq q_1(\vec{x})$ and they continue alternately, until they reach the end of the quantifiers. At the end, if $G(\vec{x}, y_1, z_1, y_2, z_2, \dots)$ becomes true the second player wins and otherwise the first player is the winner. Now it is clear that the second player in the game $\mathcal{G}_C$ has a winning strategy iff the formula C is true. With this game interpretation, any reduction program from B to A is nothing but a reduction to transfer a second player's winning strategy in the game $\mathcal{G}_A$ to a winning strategy for him in the game $\mathcal{G}_B$. Note that unlike the usual deterministic reductions between the games, these reduction programs provide a complicated protocol to transfer the winning strategies.

In the following, we will illuminate the notion of a reduction program by some concrete examples.

Example 3.5. *(Deterministic Game Reductions)* The usual complexity theoretic reduction between k-turn games is a specific example of the reduction

programs. To be more precise, assume that $\mathcal{L} = \mathcal{L}_{\rm PV}$ and $\mathcal{B} = Th(\mathbb{N})$. Now consider

$$\forall y_1 \leq p_1(\vec{x}) \exists z_1 \leq q_1(\vec{x}) \forall y_2 \leq p_2(\vec{x}) \ldots G(\vec{x}, y_1, z_1, y_2, z_2, \ldots)$$

and

$$\forall u_1 \leq p'_1(\vec{x}) \exists v_1 \leq q'_1(\vec{x}) \forall u_2 \leq p'_2(\vec{x}) \ldots H(\vec{x}, u_1, v_1, u_2, v_2, \ldots)$$

with k-many bounded quantifiers and define the following natural PV-reduction program:

[**Read** $u_1 \leq p'_1(\vec{x})$]; [**Compute** y_1 **by** $f_1(\vec{x}, u_1) \leq p_1(\vec{x})$]; [**Read** $z_1 \leq q_1(\vec{x})$]; [**Compute** v_1 **by** $g_1(\vec{x}, u_1, z_1) \leq q'_1(\vec{x})$]; ...

where f_i's and g_j's are polynomial time computable functions represented by terms in the language $\mathcal{L}_{\rm PV}$. These reductions that we call *deterministic reductions between k-turn games* are the simplest example of reduction programs.

Example 3.6. *(Non-determinism)* Let $\mathcal{L}$ be the language of PV and let $A(x, y, z)$ be an atomic formula in this language. Now consider the formulas

$$\exists u \leq s(x) \forall v \leq t(x)\ A(x, u, v)$$

and

$$\exists y y' \leq s(x) \forall z z' \leq t(x)\ [A(x, y, z) \vee A(x, y', z')]$$

in this language. Since these formulas are logically equivalent, it seems quite reasonable to assume that the first formula is reducible to the second one. Moreover, this equivalence is quite elementary and it is just on the level of pure first order logic. Hence, we can expect a very low complexity reduction in this case. Let us try to construct such a possible reduction. (Note that the notion of a reduction program is defined for formulas in prenex bounded form and since the formula $[A(x, y, z) \vee A(x, y', z')]$ is not atomic, speaking of reduction programs in this case is not technically correct. However, this is not a serious issue since we can represent the formula $A(x, y, z)$ by $\alpha(x, y, z) = 0$ for some term α and hence the formula $[A(x, y, z) \vee A(x, y', z')]$ can be safely replaced by $\alpha(x, y, z) \cdot \alpha(x, y', z') = 0$. Having all said, we still prefer keeping the original non-atomic form to be more explanatory in our discussion on the non-deterministic nature of reductions.) To construct a reduction, we have to take a look at a proof of

$$\exists u \leq s(x) \forall v \leq t(x)\ A(x, u, v)$$

from

$$\exists yy' \leq s(x) \forall zz' \leq t(x)\ [A(x,y,z) \vee A(x,y',z')].$$

The simplest proof works as the following: Assume we have $y \leq s(x)$ and $y' \leq s(x)$ such that

$$\forall zz' \leq t(x)[A(x,y,z) \vee A(x,y',z')]$$

which implies

$$\forall z \leq t(x)A(x,y,z) \vee \forall z' \leq t(x)A(x,y',z')$$

Then there are two possibilities: If $\forall z \leq t(x)A(x,y,z)$ then pick $u = y$ and if $\neg\forall z \leq t(x)A(x,y,z)$ which implies $\forall z' \leq t(x)A(x,y',z')$, pick $u = y'$.

Simulating this argument by the usual reductions between 3-turn games, as in Example 3.5, (assume the existence of a dummy bounded universal quantifier in the leftmost part of the formulas), we observe that our computational power has to be strong enough to decide $\forall z \leq t(x)A(x,y,z)$ to provide such a witness. But since $\forall z \leq t(x)A(x,y,z)$ can be extremely complex, CoNP-complete for instance, this task is far beyond the usual low complexity power that we can afford. Hence, it seems that finding a deterministic reduction is not that easy, if not impossible.

Now let us relax the strict structure of the deterministic game reductions to the following weaker non-determinism appeared in the reduction programs: In the process of witnessing, allow reductions to provide possibly more than one candidate and expect at least one of them works at a time. For instance, in this example, provide two different guesses for u like $g(x,y,y') = y$ and $h(x,y,y') = y'$ and expect the sequent

$$\{\forall v^0 \leq t(x)A(x,g(x,y,y'),v^0), \forall v^1 \leq t(x)A(x,h(x,y,y'),v^1)\}$$

to be reducible to

$$\forall z \leq t(x) \forall z' \leq t(x)\ [A(x,y,z) \vee A(x,y',z')]$$

via a PV-reduction program. For the latter, it is just enough to use the universal quantifiers to witness themselves via identity terms. More formally:

[**Read** $y \leq s(x)$]; [**Read** $y' \leq s(x)$]; [**Compute** u^0 **by** $g(x,y,y') \leq s(x)$]; [**Compute** u^1 **by** $h(x,y,y') \leq s(x)$]; [**Read** $v^0 \leq t(x)$]; [**Read** $v^1 \leq t(x)$]; [**Compute** z **by** $v^0 \leq t(x)$]; [**Compute** z' **by** $v^1 \leq t(x)$].

Hence, we can observe that non-determinism possibly provides more reductions than what the strict determinism can do. Moreover, note that this type of non-determinism is just the computational incarnation of the contraction rule which makes its use somehow unavoidable.

Example 3.7. *(Student-Teacher Game)* The real power of the reduction programs lies in the combination of non-determinism and interaction. In the Example 3.6, we observed the impact of the non-determinism part. In this example we will explain how this non-determinism leads to some complicated interactions. For this purpose, let us interpret the teacher-student game of the KPT theorem ([6]) as an example of our reduction programs. Assume we have the formula $\exists y \leq t(x) \forall z \leq s(x) A(x, y, z)$ where A is an atomic formula in the language of PV. Then consider the following PV-reduction program from $\exists y \leq t(x) \forall z \leq s(x) A(x, y, z)$ to $\top$ with length $2l$:

[**Compute** $y = y^0$ **by** $f_0(x) \leq t(x)$]; [**Read** $z = z^0 \leq s(x)$]; [**Compute** y^1 **by** $f_1(x, z^0) \leq t(x)$]; [**Read** $z^1 \leq s(x)$]; [**Compute** y^2 **by** $f_2(x, z^0, z^1) \leq t(x)$]; ...

Since it is a reduction program, the following is provable in PV:

$$\forall z^0 \dots z^{l-1} \leq s(x)[A(x, f_0(x), z^0) \vee A(x, f_1(x, z^0), z^1) \vee \dots \vee A(x, f_l(x, z^0, \dots, z^{l-1}))]$$

The point in the interaction between the so-called student and teacher is mimicked by computing y as y^0, reading $z = z^0$, computing y again under the name y^1 but this time with access to z^0, reading z^1 and computing y again under the name y^2 but now with more information about both z^0 and z^1 and so on. This non-determinism that lets us compute a variable finite many times with different functions and the interaction with the inside universal quantifiers to guess the existential quantifier again is the main power of reduction programs.

Example 3.8. *(Impossibility of Simulation)* In this example we want to provide an evidence for what we observed in the Example 3.6 to show that it is generally impossible to simulate the non-deterministic reductions and reduction programs by usual deterministic reductions. Assume $A(x, y, z, t) = (y = 0 \wedge B(x, t)) \vee (y = 1 \wedge \neg B(x, z))$ where $B(x, t)$ is an arbitrary atomic formula in the language of PV. We want to show that there is no deterministic $Th(\mathbb{N})$-reduction from

$$\exists u \leq 1 \exists v \leq s \forall w \leq s \; A(x, u, v, w)$$

to

$$\exists y y' \leq 1 \exists t t' \leq s \forall z z' \leq s \; [A(x, y, z, t) \vee A(x, y', z', t')]$$

(Note that again, our formulas are not in the prenex bounded form and hence speaking of reduction programs is not technically correct. However, we can resolve the issue as in the Example 3.6.) Assume that there exists such a deterministic $Th(\mathbb{N})$-reduction. Hence, there is a polytime function f such that:

$$\exists v \leq s \forall w \leq s\ A(x, f(x, y, y'), v, w)$$

is reducible to

$$\exists tt' \leq s \forall zz' \leq s[A(x, y, z, t) \vee A(x, y', z', t')]$$

which means that

$$\exists tt' \leq s \forall zz' \leq s[A(x, y, z, t) \vee A(x, y', z', t')]$$

implies

$$\exists v \leq s \forall w \leq s\ A(x, f(x, y, y'), v, w)$$

in $Th(\mathbb{N})$ for all $y, y' \leq 1$. Pick $y = 0$ and $y' = 1$. It is easy to see that the left side of the implication is true because either $\exists t \leq s\ B(x, t)$ or $\forall z' \leq s\ \neg B(x, z')$ is true, hence the right side should be true, as well. But the truth of the right side means

$$(f(x, 0, 1) = 0 \wedge \exists v \leq s\ B(x, v)) \vee (f(x, 0, 1) = 1 \wedge \forall w \leq s\ \neg B(x, w))$$

which means that we have a polytime decision procedure for the NP predicate $\exists w \leq s\ B(x, v)$ which implies $\mathbf{NP} = \mathbf{P}$.

Remark 3.9. The Example 3.8 shows that pure logical deductions are far beyond the power of low level deterministic reductions. In other words, it is possible to prove B by A just by some elementary methods of logic but it does not mean that B can be deterministically reducible to A. Let us explain where the problem is. At the first glance, it seems that all logical rules are completely syntactical and amenable to low complexity reductions. It is correct everywhere except for one logical rule: the contraction rule which is more or less responsible for all kinds of computational explosions like the explosion of the lengths of the proofs after the elimination of cuts. Notice that the reason that we have the equivalence in the Example 3.6 is this contraction rule and it is easy to see that this rule is the source of non-determinism and hence interactions. Therefore, it seems natural to use non-deterministic reductions to simulate computationally what is going on in the realm of proofs.

Now it is time to relate the proof theoretical non-deterministic reductions to the computational reduction programs. This is the task of our reinterpretation of the generalized Herbrand theorem for the bounded domain:

Theorem 3.10. *Let $\mathcal{B} \supseteq \mathcal{R}$ be a universal theory and $A(\vec{x})$ and $B(\vec{x})$ two formulas in the prenex bounded form. Then $A(\vec{x}) \geq_n^{\mathcal{B}} B(\vec{x})$ iff there exists a $\mathcal{B}$-reduction program from $B(\vec{x})$ to $A(\vec{x})$.*

Proof. The proof is based on the fact that any $\mathcal{B}$-reduction program is nothing but a backward interpretation of a proof that consists of some bounded quantifier rules applied on top of a quantifier-free $\mathcal{B}$-provable statement. This backward interpretation transforms the rules $(\forall^{\leq}R)$ and $(\exists^{\leq}L)$ to **Read** instructions and rules $(\exists^{\leq}R)$ and $(\forall^{\leq}L)$ to **Compute** instructions. The rest of this proof is the formalization of this very idea.

1. First assume that there exists a $\mathcal{B}$-reduction program $\{P_r\}_{r=0}^{l}$ from $B(\vec{x})$ to $A(\vec{x})$. We have to prove the following claim:

Claim. We want to show that all the free variables of $S(P^{<k})$ are among $\vec{x}$ or the variables that had been read before k. We prove the claim by induction on k. For $k = 0$ the claim is clear. For $k+1$, if P_k is the instruction $[\textbf{Read } X \leq t(\vec{x})]$, then by definition the free variables of $S(P^{<k+1})$ are among the free variables of $S(P^{<k})$ and X. By IH, all free variables of $S(P^{<k})$ had been read before k and X itself has been read in the stage k, which complete the proof. If P_k is the instruction $[\textbf{Compute } Y \textbf{ by } s \leq t(\vec{x})]$, then by definition the free variables of $S(P^{<k+1})$ are among the free variables of $S(P^{<k})$ and the free variables of s. But the free variables of s are among $\vec{x}$ and the variables that had been read before k which is exactly what we wanted to prove.

Now let us come back to prove the theorem. Define $\bar{S}(P^{<k})$ as $\forall \vec{X} \leq \vec{r}(\vec{x})[\bigwedge S^p(P^{<k}) \to \bigvee S^s(P^{<k})]$, where $\vec{X}$ are all the read variables occurred freely in $S(P^{<k})$, $\vec{r}(\vec{x})$ are their corresponding bounds and $S^p(P^{<k})$ and $S^s(P^{<k})$ are the precedent and succedent of $S(P^{<k})$, respectively. By induction on k we will show that $\mathcal{B} \vdash \bar{S}(P^{<l+1-k})$. For $k = 0$ we have the claim from the definition of a program. To prove the claim for $k+1$, we have two possibilities: First if P_{l-k} is the instruction $[\textbf{Read } X \leq t(\vec{x})]$, then $S(P^{<l-k+1})$ is defined from $S(P^{<l-k})$ by replacing all instances of $\forall X \leq t(\vec{x})A(X)$ by $A(X)$ in the right hand-side or $\exists X \leq t(\vec{x})A(X)$ by $A(X)$ in the left hand-side. Since any quantifier appears at most once, this formula is unique. On the other hand, since X is read in the stage $l-k$, then by the claim, $S(P^{<l-k})$ does not have a free variable X. By IH, $\mathcal{B} \vdash \bar{S}(P^{<l-k+1})$. Hence, we can introduce the universal bounded quantifier to have $\mathcal{B} \vdash \bar{S}(P^{<l-k})$. If P_{l-k} is the instruction $[\textbf{Compute } Y \textbf{ by } s \leq t(\vec{x})]$, then $S(P^{<l-k+1})$ is defined from $S(P^{<l-k})$ by adding $A(s)$ to its right hand-side if there is $\exists Y \leq t(\vec{x})A(Y)$

also in the right hand-side of $S(P^{<l-k})$ or by adding $A(s)$ in the left hand-side of $S(P^{<l-k})$ if $\forall Y \leq t(\vec{x})A(Y)$ is also appeared in its left hand-side. By IH, $\mathcal{B} \vdash \bar{S}(P^{<l-k+1})$. Since $\mathcal{B} \vdash s \leq t(\vec{x})$ we have $\mathcal{B} \vdash \bar{S}(P^{<l-k})$ by the introduction of bounded existential quantifier rules.

Now by induction we can conclude $\mathcal{B} \vdash S(P^{<0}) = [A(\vec{x}) \Rightarrow B(\vec{x})]$ which is what we wanted to prove.

2. For the other direction of the theorem, first note that $\mathcal{B}$ is a universal theory. Therefore, it is possible to develop a $G3$-style calculus for it, by some axioms like $P_1, P_2, \ldots, P_n \Rightarrow Q_1, Q_2, \ldots, Q_m$ where P_i's and Q_j's are atomic formulas. By Theorem 2.10 and since $\mathcal{B} \vdash A \to B$ there is a proof for the sequent $A \Rightarrow B$ in which one of the premises of any cut is an axiom. Therefore, since A and B are two formulas in the prenex bounded form, all the rules in the proof will be $G3$-style bounded quantifier rules, axioms and cuts with axioms as one of their premises. Now change the name of the variables in a way that any variable can be occurred in a quantifier at most once, and for this purpose use only the bounded variables of $A \to B$ with their possibly different variants with upper indices. More precisely, it is enough to modify the rules in the proof such that the usual $(\exists^{\leq} R)$ and $(\forall^{\leq} L)$ rules change to the following rules:

$$\frac{\Gamma, A(s), \forall y^{+1} \leq t(\vec{x})\ A^{+1}(y^{+1}) \Rightarrow \Delta}{\Gamma, s \leq t(\vec{x}), \forall y \leq t(\vec{x})\ A(y) \Rightarrow \Delta} \qquad \frac{\Gamma \Rightarrow \Delta, A(s), \exists y^{+1} \leq t(\vec{x})\ A^{+1}(y^{+1})}{\Gamma, s \leq t(\vec{x}), \Rightarrow \Delta, \exists y \leq t(\vec{x})\ A(y)}$$

where C^{+1} means increasing the upper index of any bounded variable in C by one and y^{+1} means increasing the upper index of y by one. Then since any variable occurs at most in one quantifier, we can change all the rules $(\exists^{\leq} L)$ and $(\forall^{\leq} R)$ to:

$$\frac{\Gamma, y \leq t \Rightarrow \Delta, A(y)}{\Gamma \Rightarrow \Delta, \forall y \leq t\ A(y)} \qquad \frac{\Gamma, y \leq t, A(y) \Rightarrow \Delta}{\Gamma, \exists y \leq t\ A(y) \Rightarrow \Delta}$$

in a way that all the sequents in the proof remain $\mathcal{B}$-provable. The main point is that if any variable occurs at most in one quantifier, substituting the eigenvariable b in the rules $(\exists^{\leq} L)$ and $(\forall^{\leq} R)$ by the bounding variable y itself, does not affect the validity of the proof.

Now, by induction on the length of the proof, we will show that if $\Gamma \Rightarrow \Delta$ appears in a stage of this proof, then there is a $\mathcal{B}$-reduction-program $P = \{P_r\}_{r=0}^{l}$ with $S(P^{<0}) = (\Gamma \Rightarrow \Delta)$ using exactly the variables in the proof

with the condition that the variable z^m becomes a variant of the variable z^n when $m > n$. (Note that this condition is inconsistent with our naming condition in the definition of the reduction programs which states that the variable z^k should be considered as a variant of z^0 when $k > 0$ and z^0 occurs as a bounded variable in $S(P^{<0})$. However, this is just a change in the names of the variables that makes everything simpler. Therefore, the rest of this proof should be read, up to this change in the naming condition.)

The claim for the axioms is straightforward. For the bounded existential rule, assume that $\Gamma, s \leq t(\vec{x}) \Rightarrow \Delta, \exists y \leq t(\vec{x})A(y)$ is a consequence of $\Gamma \Rightarrow \Delta, A(s), \exists y^{+1} \leq t(\vec{x})A^{+1}(y)$. Then by IH, there exists a program P with the condition that $S(P^{<0}) = (\Gamma \Rightarrow \Delta, A(s), \exists y^{+1} \leq t(\vec{x})A^{+1}(y))$. Define s' as:

$$s' = \begin{cases} s & \text{if } s \leq t(\vec{x}) \\ t(\vec{x}) & \text{if } s > t(\vec{x}) \end{cases}$$

It is possible to find such a term because the language is powerful enough to have the characteristic function for the order predicate as observed in Remark 2.2. Moreover, note that $\mathcal{B} \vdash s' \leq t(\vec{x})$. Define $P' = P$ with different initial sequent $S(P'^{<0})$ as $(\Gamma, s \leq t(\vec{x}) \Rightarrow \Delta, A(s'), \exists y^{+1} \leq t(\vec{x})A^{+1}(y))$. The reason that P' is also a reduction program is the following: The sequent $S(P^{<l+1})$ has a quantifier-free $\mathcal{B}$-provable subsequent S'. But the difference between $S(P^{<l+1})$ and $S(P'^{<l+1})$ is in adding the formula $s \leq t(\vec{x})$ in the left hand-side of $S(P^{<l+1})$ and substituting s' for s in A. We know that $\mathcal{B} \vdash s \leq t(\vec{x}) \rightarrow s = s'$. Pick the correspondent of the S' in $S(P'^{<l+1})$ (S' after substitution s' for s) and call it $S'' = \Gamma'' \Rightarrow \Delta''$. Hence, $\mathcal{B} \vdash \Gamma'', s \leq t(\vec{x}) \Rightarrow \Delta''$ which implies that P' is a reduction program.

Now define Q by $Q_r = P'_{r+1}$ for $r \leq l$ and Q_0 as the instruction [**Compute** y **by** $s' \leq t(\vec{x})$] and $S(Q^{<0}) = (\Gamma, s \leq t(\vec{x}) \Rightarrow \Delta, \exists y \leq t(\vec{x})A(y))$. It is pretty clear that Q is a reduction program which proves the claim.

A similar argument also works for the bounded universal quantifier rule. The only case that we have to check is the cut rule. Assume $(\Gamma, \vec{P} \Rightarrow \Delta, \vec{Q})$ is a consequence of $(\Gamma, \vec{P}, R \Rightarrow \Delta, \vec{Q})$ and $(\Gamma, \vec{P} \Rightarrow R, \vec{Q}, \Delta)$ where the first is an instance of an axiom with the main sequent $(\vec{P}, R \Rightarrow \vec{Q})$ and the second is provable. By IH there exists a program reducing $\{R, \vec{Q}, \Delta\}$ to $\{\Gamma, \vec{P}\}$. This program also essentially works for reducing $\{\vec{Q}, \Delta\}$ to $\{\Gamma, \vec{P}\}$. More precisely, define $P' = P$ with different initial sequent as $S(P'^{<0}) = (\{\Gamma, \vec{P}\} \Rightarrow \{\vec{Q}, \Delta\})$. The only important thing is showing that $S(P'^{<l+1})$ has a quantifier-free $\mathcal{B}$-provable subsequent. From IH we know that there exists a quantifier-free

$\mathcal{B}$-provable subsequent of $S(P^{<l+1})$ which we call $S' = (\Gamma' \Rightarrow \Delta')$. Since all $\vec{P}$, $\vec{Q}$ and R are atomic formulas, they will remain intact through the quantifier opening process of the reduction program, hence the difference between $S(P'^{<l+1})$ and $S(P^{<l+1})$ is in one instance of R in the right-hand side of $S(P^{<l+1})$. Moreover, it implies that $(\mathbf{P} \Rightarrow \mathbf{Q})$ is a subsequent of both $S(P^{<l+1})$ and $S(P'^{<l+1})$. Define $S'' = (\Gamma' \Rightarrow \Delta' - \{R\})$. We show that S'' is a $\mathcal{B}$-provable quantifier-free subsequent of $S(P'^{<l+1})$. Since $\vec{P} \subseteq \Gamma'$ and $\vec{Q}, R \subseteq \Delta'$ we have $\mathcal{B} \vdash \Gamma', R \Rightarrow \Delta' - \{R\}$ because it is an instance of the axioms. Since $\mathcal{B} \vdash \Gamma' \Rightarrow \Delta'$ by cut we have $\mathcal{B} \vdash S''$ which completes the proof.

□

4 Non-deterministic Flows

In the last section, we defined the concept of a reduction which can be considered as a one-step move of the computational content. Now it is time to let it *flow*:

Definition 4.1. Let Π be a π-class, $A(\vec{x}), B(\vec{x}) \in \Pi$, $\mathcal{B} \supseteq \mathcal{R}$ a theory and $\mathbb{T}$ a $\mathcal{B}$-term ideal. A non-deterministic $(\mathbb{T}, \Pi, \mathcal{B})$-flow from $A(\vec{x})$ to $B(\vec{x})$ is a pair (t, H) where $t(\vec{x}) \in \mathbb{T}$ is a term and $H(u, \vec{x}) \in \Pi$ is a formula such that the following statements are provable in $\mathcal{B}$:

(i) $H(0, \vec{x}) \leftrightarrow A(\vec{x})$.

(ii) $H(t(\vec{x}), \vec{x}) \leftrightarrow B(\vec{x})$.

(iii) $\forall u < t(\vec{x})\ H(u, \vec{x}) \rightarrow H(u+1, \vec{x})$.

If there exists a non-deterministic $(\mathbb{T}, \Pi, \mathcal{B})$-flow from $A(\vec{x})$ to $B(\vec{x})$ we will write $A(\vec{x}) \rhd_n^{(\mathbb{T},\Pi,\mathcal{B})} B(\vec{x})$. Moreover, if Γ and Δ are sequents of formulas in Π, by $\Gamma \rhd_n^{(\mathbb{T},\Pi,\mathcal{B})} \Delta$ we mean $\bigwedge \Gamma \rhd_n^{(\mathbb{T},\Pi,\mathcal{B})} \bigvee \Delta$. The case for $(\mathbb{T}, \Sigma, \mathcal{B})$-flows is defined similarly by changing Π everywhere with Σ.

Convention. In the remaining part of this section, we will fix an arbitrary choice for the type of a flow as $(\mathbb{T}, \Sigma, \mathcal{B})$-flow or $(\mathbb{T}, \Pi, \mathcal{B})$-flow. For simplicity, and to address both cases simultaneously, we will use the letters Φ and ϕ, standing for a fixed choice from two cases $[\Phi = \Sigma$ and $\phi = \sigma]$ or $[\Phi = \Pi$ and $\phi = \pi]$. For instance, by the sentence "Φ is a ϕ-class" we mean either "Σ is a σ-class" or "Π is a π-class". Moreover, we use the shorthand $\rhd$ for $\rhd_n^{(\mathbb{T},\Phi,\mathcal{B})}$ for simplicity and if emphasis on some parts of the triple $(\mathbb{T}, \Phi, \mathcal{B})$

becomes needed, we put back those parts as the superscript of $\rhd$. For instance, if we write $\rhd^{\Phi}$, we want to emphasize on the class of the flow.

The following theorem is the main theorem of the theory of non-deterministic flows for bounded theories of arithmetic.

Theorem 4.2. *Let Φ be a ϕ-class, $\Gamma(\vec{x})\cup\Delta(\vec{x}) \subseteq \Phi$ and $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathfrak{B}(\mathbb{T}, \Phi, \mathcal{A})$. Then $\mathfrak{B}(\mathbb{T}, \Phi, \mathcal{A}) \vdash \Gamma(\vec{x}) \Rightarrow \Delta(\vec{x})$ iff $\Gamma \rhd_n^{(\mathbb{T},\Phi,\mathcal{B})} \Delta$.*

To prove this theorem we need the following sequence of lemmas. These lemmas provide a high level calculus for the relation $\rhd$ which makes its use more effective in any practical situation.

Lemma 4.3. *(i) (Weak Gluing) If $A(\vec{x}) \rhd B(\vec{x})$ and $B(\vec{x}) \rhd C(\vec{x})$ then $A(\vec{x}) \rhd C(\vec{x})$.*

(ii) (Strong Gluing) If $A(y, \vec{x}) \rhd A(y+1, \vec{x})$ and $s \in \mathbb{T}$, then $A(0, \vec{x}) \rhd A(s, \vec{x})$.

Proof. For (i) since $A(\vec{x}) \rhd_n B(\vec{x})$ there exists a term $t(\vec{x}) \in \mathbb{T}$, a formula $H(u, \vec{x}) \in \Phi$ such that $\mathcal{B}$ proves the conditions in the Definition 4.1. On the other hand since $B(\vec{x}) \rhd_n C(\vec{x})$ we have the corresponding data for $B(\vec{x})$ to $C(\vec{x})$ which we show by $t'(\vec{x})$ and $H'(u, \vec{x})$. Define $r(\vec{x}) = t(\vec{x}) + t'(\vec{x}) + 1$ and

$$I(u, \vec{x}) = \begin{cases} H(u, \vec{x}) & \text{if } u \leq t(\vec{x}) \\ B(\vec{x}) & \text{if } u = t(\vec{x}) + 1 \\ H'(u \dot{-} t(\vec{x}) \dot{-} 2, \vec{x}) & \text{if } t(\vec{x}) + 1 < u \leq t(\vec{x}) + t'(\vec{x}) + 1 \end{cases}$$

Then, it is easy to check that this new data is a non-deterministic $(\mathbb{T}, \Phi, \mathcal{B})$-flow from $A(\vec{x})$ to $C(\vec{x})$. Notice that since $\mathbb{T}$ is closed under successor and addition and $t, t' \in \mathbb{T}$, we have $r \in \mathbb{T}$.

For (ii), if we have $A(y, \vec{x}) \rhd_n A(y + 1, \vec{x})$ it is enough to glue all copies of the sequences of reductions for $0 \leq y \leq s$, to have $A(0, \vec{x}) \rhd_n A(s, \vec{x})$. More precisely, assume that all reductions have the same length $t'(\vec{x})$ greater than $t(s, \vec{x})$. This is an immediate consequence of the facts that we can find a monotone majorization for $t(y, \vec{x})$ like $r(y, \vec{x})$, and since $y \leq s$ we have $t(y, \vec{x}) \leq r(y, \vec{x}) \leq r(s, \vec{x})$. Now it is enough to repeat the last formula in the flow to make the flow longer to reach the length $t'(\vec{x}, \vec{z}) = r(s, \vec{x})$ where $\vec{z}$ is a vector of variables in s. Now, define $t''(\vec{x}, \vec{z}) = s \times (t'(\vec{x}) + 2)$,

$$I(u, \vec{x}) = \begin{cases} H(u, y, \vec{x}) & \text{if } y(t' + 2) < u < (y + 1)(t' + 2) \\ A(y, \vec{x}) & \text{if } u = y(t' + 2) \end{cases}$$

and

$$F(u) = \begin{cases} F(u,y) & \text{if } y(t'+2) < u < (y+1)(t'+2) \dot{-} 1 \\ E_0(u,y) & \text{if } u = y(t'+2) \\ G_1(u,y+1) & \text{if } u = (y+1)(t'+2) \dot{-} 1 \end{cases}$$

It is easy to see that this new sequence is a non-deterministic $(\mathbb{T}, \Phi, \mathcal{B})$-flow from $A(0,\vec{x})$ to $A(s,\vec{x})$. Notice that $\mathbb{T}$ is closed under substitution, sum and product and therefore, $t'' \in \mathbb{T}$. □

Lemma 4.4. *(Conjunction and Disjunction Rules)*

(*i*) *If* $\Gamma, A \rhd \Delta$ *or* $\Gamma, B \rhd \Delta$ *then* $\Gamma, A \wedge B \rhd \Delta$.

(*ii*) *If* $\Gamma_0 \rhd \Delta_0, A$ *and* $\Gamma_1 \rhd \Delta_1, B$ *then* $\Gamma_0, \Gamma_1 \rhd \Delta_0, \Delta_1, A \wedge B$.

(*iii*) *If* $\Gamma \rhd \Delta, A$ *or* $\Gamma \rhd \Delta, B$ *then* $\Gamma \rhd \Delta, A \vee B$.

(*iv*) *If* $\Gamma_0, A \rhd \Delta_0$ *and* $\Gamma_1, B \rhd \Delta_1$ *then* $\Gamma_0, \Gamma_1, A \vee B \rhd \Delta_0, \Delta_1$.

Proof. (*i*) and (*iii*), are trivial simply because firstly we have $A \wedge B \geq A$, $A \wedge B \geq B$, $A \geq A \vee B$ and $B \geq A \vee B$ and then we can add the needed formula in the beginning or the end of the flow.

For (*ii*) and (*iv*), we will prove (*ii*), (*iv*) is just dual to (*ii*). If $\Gamma_0 \rhd \Delta_0, A$, then clearly we have $\bigwedge \Gamma_0 \wedge \bigwedge \Gamma_1 \rhd (\bigvee \Delta_0 \vee A) \wedge \bigwedge \Gamma_1$. Moreover, we have $\bigwedge \Gamma_1 \rhd \bigvee \Delta_1 \vee B$ and again we have $\bigwedge \Gamma_1 \wedge (\bigvee \Delta_0 \vee A) \rhd (\bigvee \Delta_1 \vee B) \wedge (\bigvee \Delta_0 \vee A)$. Therefore by weak gluing

$$\bigwedge \Gamma_0 \wedge \bigwedge \Gamma_1 \rhd (\bigvee \Delta_1 \vee B) \wedge (\bigvee \Delta_0 \vee A).$$

But it is easy to see that

$$(\bigvee \Delta_1 \vee B) \wedge (\bigvee \Delta_0 \vee A) \geq_n \bigvee \Delta_1 \vee \bigvee \Delta_0 \vee (A \wedge B).$$

Hence

$$\Gamma_0, \Gamma_1 \rhd \Delta_0, \Delta_1, (A \wedge B).$$

□

In the following, wherever we write $\neg A$, we mean any possible formula B such that $\vdash \neg A \leftrightarrow B$.

Lemma 4.5. *(Negation Rules) If* $\Gamma, \Delta \subseteq \Phi$ *and* $A, \neg A \in \Phi$ *then*

(*i*) *If* $\Gamma, A \rhd^{\Phi} \Delta$ *then* $\Gamma \rhd^{\Phi} \Delta, \neg A$.

(*ii*) *If* $\Gamma \rhd^{\Phi} \Delta, A$ *then* $\Gamma, \neg A \rhd^{\Phi} \Delta$.

Proof. We will prove (*i*), (*ii*) is similar. Since $\Gamma, A \rhd^{\Phi} \Delta$ there exists $t \in \mathbb{T}$ and $H \in \Phi$ such that the conditions of the Definition 4.1 hold. Now, use $H \wedge \neg A$ as the formula to have a flow from $(\bigwedge \Gamma \wedge A) \vee \neg A$ to $\bigvee \Delta \vee \neg A$. Since

$$\mathcal{B} \vdash \bigwedge \Gamma \to (\bigwedge \Gamma \wedge A) \vee \neg A$$

by adding $\bigwedge \Gamma$ to the beginning of the flow we have a flow from Γ to $\Delta, \neg A$. □

Remark 4.6. Note that the cut and induction rules are derivable in the presence of the structural and propositional rules and their context-free versions, i.e,

$$\frac{A \Rightarrow B \qquad B \Rightarrow C}{A \Rightarrow C} \qquad \frac{A(y) \Rightarrow A(y+1)}{A(0) \Rightarrow A(t)}$$

Therefore since we have weak and strong gluing lemmas, we do not need to prove cut and induction in a separate lemma.

Lemma 4.7. *(Implication Rules) If* $A \to B \in \Phi$*:*

(*i*) *If* $\Gamma_0 \rhd^{\Phi} \Delta_0, A$ *and* $\Gamma_1, B \rhd^{\Phi} \Delta_1$ *then* $\Gamma_0, \Gamma_1, A \to B \rhd^{\Phi} \Delta_0, \Delta_1$.

(*ii*) *If* $\Gamma, A \rhd^{\Phi} \Delta, B$ *then* $\Gamma \rhd^{\Phi} \Delta, A \to B$.

Proof. Note that when $A \to B \in \Pi$ then since Π is closed under subformulas, we have $A, B \in \Pi$. For (*i*), since $\Gamma_0 \rhd \Delta_0, A$ by applying conjunction with $A \to B$ everywhere in the flow, we have

$$\bigwedge \Gamma_0 \wedge (A \to B) \rhd (\bigvee \Delta_0 \vee A) \wedge (A \to B).$$

Since

$$(\bigvee \Delta_0 \vee A) \wedge (A \to B) \rhd \bigvee \Delta_0 \vee (A \wedge (A \to B)),$$

and $A \wedge A \to B \geq_n B$, we have

$$\bigvee \Delta_0 \vee (A \wedge (A \to B)) \rhd \bigvee \Delta_0 \vee B.$$

And then since $\Gamma_1 \rhd B, \Delta_1$, by cut on B we have

$$\Gamma_0, \Gamma_1, A \to B \rhd \Delta_0, \Delta_1.$$

For (ii), if $\Gamma, A \rhd B, \Delta$, then by applying disjunction with $A \to B$ everywhere in the flow,

$$(\bigwedge \Gamma \wedge A) \vee (A \to B) \rhd \bigvee \Delta \vee B \vee (A \to B).$$

And since

$$((\bigwedge \Gamma \vee (A \to B)) \wedge (A \vee (A \to B)) \rhd (\bigwedge \Gamma \wedge A) \vee (A \to B),$$

we have

$$((\bigwedge \Gamma \vee (A \to B)) \wedge (A \vee (A \to B)) \rhd \bigvee \Delta \vee B \vee (A \to B).$$

Since $B \geq_n (A \to B)$, by contraction and cut we have $B \vee (A \to B) \rhd A \to B$. On the other hand, $\geq A \vee (A \to B)$. Hence

$$\Gamma \rhd ((\bigwedge \Gamma \vee (A \to B)) \wedge (A \vee (A \to B)),$$

and therefore by gluing $\Gamma \rhd \Delta, A \to B$. □

Now we are ready to prove the following soundness theorem as the first half of the main theorem:

Theorem 4.8. *(Soundness) If Φ is a ϕ-class, $\Gamma(\vec{x}) \cup \Delta(\vec{x}) \subseteq \Phi$, $\mathfrak{B}(\mathbb{T}, \Phi, \mathcal{A}) \vdash \Gamma(\vec{x}) \Rightarrow \Delta(\vec{x})$ and $\mathcal{A} \subseteq \mathcal{B}$ then $\Gamma \rhd_n^{(\mathbb{T},\Phi,\mathcal{B})} \Delta$.*

Proof. We assume $\Phi = \Pi$ is a π-type class, the other case is similar. To prove the theorem we use induction on the length of the free-cut free proof of $\Gamma(\vec{x}) \Rightarrow \Delta(\vec{x})$. The importance of using the free-cut free proof is its usual consequence that all the formulas occurring in the proof belong to the class Π itself. (See Corollary 2.11.) Since Π consists of bounded formulas, it also implies that the only used quantifier rules are bounded quantifier rules. Hence, we have the following cases:

1. (Axioms). If $\Gamma(\vec{x}) \Rightarrow \Delta(\vec{x})$ is a logical axiom then the claim is trivial. If it is a non-logical axiom then the claim will be also trivial because all non-logical axioms are quantifier-free and provable in $\mathcal{B}$. Therefore there is nothing to prove.

2. (Structural Rules). We will prove the case of the contraction rule, the rest are similar. Assume that $\Gamma, A \Rightarrow \Delta$ is prove by left contraction from

$\Gamma, A, A \Rightarrow \Delta$. Then by IH, there exists a flow from $\bigwedge \Gamma \wedge (A \wedge A)$ to $\bigvee \Delta$. Since $\mathcal{B} \vdash A \to A \wedge A$, we know

$$\bigwedge \Gamma \wedge A \geq \bigwedge \Gamma \wedge (A \wedge A)$$

adding $\bigwedge \Gamma \wedge A$ to the beginning of the flow, we will have a flow from $\bigwedge \Gamma \wedge A$ to $\bigvee \Delta$ which proves what we wanted.

3. (Cut). See the Remark 4.6.

4. (Propositional Rules). The conjunction and disjunction cases are proved in the Lemma 4.4. The implication and negation cases are proved in the Lemmas 4.5 and 4.7, respectively.

5. (Bounded Universal Quantifier Rules, Right). If

$$\Gamma(\vec{x}) \Rightarrow \Delta(\vec{x}), \forall z \leq p(\vec{x}) B(\vec{x}, z)$$

is proved by the $\forall^{\leq} R$ rule by $\Gamma(\vec{x}), b \leq p(\vec{x}) \Rightarrow \Delta(\vec{x}), B(\vec{x}, b)$, then by IH we have $\Gamma(\vec{x}), b \leq p(\vec{x}) \rhd \Delta(\vec{x}), B(\vec{x}, b)$. Therefore, there exists a term $t(\vec{x}) \in \mathbb{T}$, a formula $H(u, \vec{x}, b) \in \Pi$ such that the conditions of the Definition 4.1 are provable in $\mathcal{B}$. First of all, extend the sequence by repeating the last formula to reach a majorization $s(\vec{x})$ as in the previous part. Then, define $t'(\vec{x}) = s(\vec{x})$ and $H'(u, \vec{x}) = \forall z \leq p(\vec{x}) H(u, \vec{x}, z)$. Since $H(u, \vec{x}, b) \in \Pi$ and Π is closed under substitution, we have $H(u, \vec{x}, z) \in \Pi$ and hence $\forall z \leq p(\vec{x}) H(u, \vec{x}, z) \in \Pi$. The other conditions to check that the new sequence is a $(\mathbb{T}, \Pi, \mathcal{B})$-flow is a straightforward consequence of the fact that if $\mathcal{B} \vdash \forall u \leq t'(\vec{x}) H(u, b, \vec{x}) \to H(u + 1, b, \vec{x})$, then

$$\mathcal{B} \vdash \forall u \leq t'(\vec{x}) \forall z \leq p(\vec{x}) H(u, z, \vec{x}) \to \forall z \leq p(\vec{x}) H(u + 1, z, \vec{x}).$$

Finally add $\bigwedge \Gamma$ to the beginning of the flow and add $\forall z \leq p(\vec{x}) B(\vec{x}, z) \vee \bigvee \Delta$ to its end, then the new flow would be a flow from Γ to $\Delta(\vec{x}), \forall z \leq p(\vec{x}) B(\vec{x}, z)$. Note that the new length is constructed by majorizing, substitution and successor from $t \in \mathbb{T}$, hence it is also in $\mathbb{T}$.

6. (Bounded Universal Quantifier Rules, Left). Suppose $\Gamma(\vec{x}), s(\vec{x}) \leq p(\vec{x}), \forall z \leq p(\vec{x}) B(\vec{x}, z) \Rightarrow \Delta(\vec{x})$ is proved by the $\forall^{\leq} L$ rule by $\Gamma(\vec{x}), B(\vec{x}, s(\vec{x})) \Rightarrow \Delta(\vec{x})$. Since $\mathcal{B} \vdash s(\vec{x}) \leq p(\vec{x}) \wedge \forall z \leq p(\vec{x}) B(\vec{x}, z) \to B(\vec{x}, s(\vec{x}))$, we have

$$s(\vec{x}) \leq p(\vec{x}), \forall z \leq p(\vec{x}) B(\vec{x}, z) \geq B(\vec{x}, s(\vec{x})).$$

Since

$$\Gamma(\vec{x}), B(\vec{x}, s(\vec{x})) \rhd \Delta(\vec{x}),$$

by cut we have

$$\Gamma(\vec{x}), s(\vec{x}) \leq p(\vec{x}), \forall z \leq p(\vec{x}) B(\vec{x}, s(\vec{x})) \rhd \Delta(\vec{x}).$$

Moreover, note that t'' is constructed by majorizing, substitution and successor from $t \in \mathbb{T}$, hence $t'' \in \mathbb{T}$.

7. (Bounded Existential Quantifier Rules, Right). It is similar to 6.

8. (Bounded Existential Quantifier Rules, Left). If $\Gamma, \exists y \leq p(\vec{x}) B(\vec{x}, y) \Rightarrow \Delta$ is proved by the $\exists^{\leq} L$ rule by $\Gamma, b \leq p(\vec{x}), B(\vec{x}, b) \Rightarrow \Delta$, by IH we have $\Gamma, b \leq p(\vec{x}), B(\vec{x}, b) \rhd \Delta$ then since $\exists y \leq p(\vec{x}) B(\vec{x}, y) \in \Pi$, $B(\vec{x}, y)$ has a negation in Π. Since Π is closed under substitution, $B(\vec{x}, b)$ also has a negation in Π. Therefore, by Lemma 4.5

$$\Gamma, b \leq p(\vec{x}) \rhd \Delta, \neg B(\vec{x}, b)$$

by 5, we have

$$\Gamma \rhd \Delta, \forall y \leq p(\vec{x})\ \neg B(\vec{x}, y)$$

Finally again by Lemma 4.5 we have

$$\Gamma, \exists y \leq p(\vec{x}) B(\vec{x}, y) \rhd \Delta.$$

9. (Induction). See the Remark 4.6. □

We also have the following completeness theorem:

Theorem 4.9. *(Completeness) If* $\Gamma(\vec{x}) \rhd_n^{(\mathbb{T},\Phi,\mathcal{B})} \Delta(\vec{x})$ *and* $\mathcal{B} \subseteq \mathfrak{B}(\mathbb{T}, \Phi, \mathcal{A})$, *then* $\mathfrak{B}(\mathbb{T}, \Phi, \mathcal{A}) \vdash \Gamma(\vec{x}) \Rightarrow \Delta(\vec{x})$.

Proof. If $\Gamma(\vec{x}) \rhd_n^{(\mathbb{T},\Pi,\mathcal{B})} \Delta(\vec{x})$, then by Definition 3.1, there exist a term $t(\vec{x}) \in \mathbb{T}$, and a formula $H(u, \vec{x}) \in \Pi$ such that we have the following:

(*i*) $\mathcal{B} \vdash H(0, \vec{x}) \leftrightarrow \bigwedge \Gamma(\vec{x})$,

(*ii*) $\mathcal{B} \vdash H(t(x), \vec{x}) \leftrightarrow \bigvee \Delta(\vec{x})$,

(*iii*) $\mathcal{B} \vdash \forall u \leq t(\vec{x})\ H(u, \vec{x}) \rightarrow H(u+1, \vec{x})$.

Since $\mathcal{B} \subseteq \mathfrak{B}(\mathbb{T}, \Pi, \mathcal{A})$, we have

$$\mathfrak{B}(\mathbb{T}, \Pi, \mathcal{A}) \vdash \forall u \leq t(\vec{x})\ H(u, \vec{x}) \rightarrow H(u+1, \vec{x}).$$

Since $H(u, \vec{x}) \in \Pi$ and $t \in \mathbb{T}$, by induction we have ,

$$\mathfrak{B}(\mathbb{T}, \Pi, \mathcal{A}) \vdash H(0, \vec{x}) \rightarrow H(t(\vec{x}), \vec{x}).$$

On the other hand, we have $\mathcal{B} \vdash H(0, \vec{x}) \leftrightarrow \bigwedge \Gamma(\vec{x})$ and $\mathcal{B} \vdash H(t(\vec{x}), \vec{x}) \leftrightarrow \bigvee \Delta(\vec{x})$. Therefore, $\mathfrak{B}(\mathbb{T}, \Pi, \mathcal{A}) \vdash \Gamma(\vec{x}) \Rightarrow \Delta(\vec{x})$. □

5 Applications

In this section we will explain some applications of the theory of non-deterministic flows. For this purpose, let us first define a hierarchy of theories of bounded arithmetic to have a variety of theories with different induction lengths for which the non-determinism is the most effective trick. For this purpose, consider the language $\mathcal{L}_n$ as the Buss' language of bounded arithmetic, [4], augmented with subtraction, division and the function symbols $\{\#_k\}_{k\leq n}$ and define BASIC_n as the theory $\mathcal{A}_p$ as in the Example 2.6, together with the defining axioms for these new function symbols. These axioms include the axioms of the theory $\mathcal{R}$ and a suitable representation of $x\#_{k+1}y = 2^{|x|\#_k|y|}$. For $m \leq n-1$, define $\mathbb{T}_{n,m}$ as the set consisting of all terms less than the terms of the form $|t|_m$, provably in BASIC_n where $|t|_m$ means applying the length function m many times. We claim that $\mathbb{T}_{n,m}$ is a BASIC_n-term ideal. First note that for any terms t and s, BASIC_n proves that $|t|_m \cdot |s|_m \leq |t\#_{m+1}s|_m$, hence by $m+1 \leq n$, it is easy to prove that $\mathbb{T}_{n,m}$ is closed under addition, multiplication, subtraction and division. Secondly, it is clear that this set is closed under substitutions simply because of its form and finally note that the set has the majorizing terms of the form $|t|_m$ where t just consists of increasing function symbols, i.e. all the function symbols excluding subtraction and division. Now, define the theory $R^k_{m,n}$ as the bounded arithmetic $\mathfrak{B}(\mathbb{T}_m, \hat{\Pi}^b_k(\#_n), \mathrm{BASIC}_n)$.

In the following theorem, we show that it is possible to decompose proofs of $R^k_{m,n}$:

Theorem 5.1. *Let* $\Gamma, \Delta \subseteq \hat{\Pi}^b_k(\#_n)$, *then* $R^k_{m,n} \vdash \Gamma \Rightarrow \Delta$ *iff*

$$\Gamma \rhd_n^{(\mathbb{T}_m, \hat{\Pi}^b_k(\#n), \mathrm{BASIC}_n)} \Delta.$$

Note that more smash functions can simulate higher order objects in our first order setting. For instance the theory $R^k_{n-1,n+r}$ can be read as a theory powerful enough to talk about the n-th order objects, has first order induction for the formulas with k-many alternations of these higher order objects and finally has the $\#_{r+1}$ function on the first order elements. For instance having a characterization of $S^k_3 = R^k_{1,3}$-provable sentences of the form $\forall x \exists y \leq |t(x)|\ A(x,y)$ where A is quantifier-free in the language augmented with all computable functions in time $|t|$ where $t \in \mathcal{L}(\#_3)$, is equivalent to providing a characterization of the total NP-search problems of the second order hierarchy V^k_2.

Using the Theorem 5.1 for the usual bounded theories S_2^k, we can provide an example of the combination of the ingredients that we have mentioned in the Introduction, i.e., first transforming a proof to a sequence of implications over a universal theory and then using the Theorem 3.10 to bring the computational content of each implication.

Corollary 5.2. *Let $A(\vec{x}), B(\vec{x}) \in \mathcal{L}_{\mathrm{PV}}$ be two formulas in the k-prenex bounded form and $S_2^k(\mathrm{PV})$ be the theory S_2^k written in the language of* PV. *Then $S_2^k(\mathrm{PV}) \vdash A(\vec{x}) \to B(\vec{x})$ iff there exists a polynomial p, a formula $G(u, \vec{x})$ in the k-prenex bounded form with bounds depending only on $\vec{x}$, a uniform sequence of* PV*-reduction programs P_u from $G(u, \vec{x})$ to $G(u+1, \vec{x})$ and two* PV*-reduction programs, one from $A(\vec{x})$ to $G(0, \vec{x})$ and the other from $G(p(|\vec{x}|), \vec{x})$ to $B(\vec{x})$.*

Proof. Since $S_2^k(\mathrm{PV})$ is axiomatizable by $(\mathbb{T}_{1,2}, \hat{\Pi}_k^b)$-induction; all quantifier-free formulas have PV-equivalent atomic representation and PV is a universal theory, the claim is a clear consequence of Theorem 5.1 and Theorem 3.10. □

Remark 5.3. Note that this theorem transforms the provability of implications of $\hat{\Pi}_k^b$ formulas (written in their k-prenex bounded forms) in S_2^k (written in the language of PV) to the existence of polynomially long sequence of k-turn games with a uniform sequence of PV-reduction programs between them. This characterization is more or less similar to the characterizations of [8] and [9] for the theories T_2^k. The difference is on the length of the sequences which in our case is polynomial and hence exponentially shorter that the exponentially long sequence of reductions of [8] and [9]. However, our reduction steps are non-deterministic and hence far more complicated than the simple deterministic reductions of [8] and [9]. Using the $\hat{\Sigma}_{k+1}^b$-conservativity of S_2^{k+1} over T_2^k, we can use both characterizations for both theories for appropriate complexity. This technique pushes the previously known characterization of $\forall\hat{\Sigma}_j^b$ consequences of T_2^k for $1 \leq j \leq k$ ([9]), one level up to provide also a characterization of $\forall\hat{\Sigma}_{k+1}^b$ consequences of T_2^k. It is also worth mentioning that we can apply our characterization to provide another combinatorial characterization of the total NP search problems of the theory S_2^{k+1}, and hence of T_2^k, based on polynomially long sequence of PV-reduction-programs.

For the second application, we propose a new proof for the strong version of witnessing theorems for the hierarchy S_2^k. This type of strong witnessing theorems appeared in [8], [3], [2], [5] and [9] for different bounded theories including the theories S_2^k.

Define the hierarchy of function classes $\Box^p_k$ as: $\Box^p_1 = \mathrm{FP}$, $\Box^p_{k+1} = \mathrm{FP}^{\Sigma^p_k}$ and let $comp(\vec{x}, M, w)$ be a polytime formalization for "*w is a computation of the algorithm M on the inputs $\vec{x}$*" and $out(w)$ be a polynomial time function symbol which reads w and computes the output of w. Then:

Corollary 5.4. *(Strong Witnessing Theorem) The provably $\hat{\Sigma}^b_k$-definable functions of S^k_2 are in $\Box^p_k$, provably in* PV*, i.e. if $S^k_2 \vdash \forall \vec{x} \exists y A(\vec{x}, y)$ where $A(\vec{x}, y) \in \hat{\Sigma}^b_k$, then there exists a machine M computing a function $f \in \Box^p_k$ such that* $\mathrm{PV} \vdash comp(\vec{x}, M, w) \to A(\vec{x}, out(w))$.

Proof. Assume $S^k_2 \vdash \forall \vec{x} \exists y A(\vec{x}, y)$. By Parikh theorem we know that there exists a bound for the existential quantifier. Hence there exists a term $t(\vec{x})$ such that $S^k_2 \vdash \forall y \leq t(\vec{x})\ \neg A(\vec{x}, y) \Rightarrow \bot$. W.l.o.g, assume that the language extends the language of PV. Hence, S^k_2 is axiomatizable in this language by $(\mathbb{T}_{1,2}, \hat{\Pi}^b_k)$-induction. By Theorem 5.1, there exist a polynomial $p(|\vec{x}|)$ and a formula $H(u, \vec{x}) \in \hat{\Pi}^b_k$ such that the following statements are provable in PV:

(i) $H(0, \vec{x}) \leftrightarrow [\forall y \leq t(\vec{x})\ \neg A(\vec{x}, y)]$.

(ii) $H(p(|\vec{x}|), \vec{x}) \leftrightarrow \bot$.

(iii) $\forall u < p(|\vec{x}|)\ H(u, \vec{x}) \to H(u+1, \vec{x})$.

W.l.o.g we can assume that H is in the k-prenex bounded form. Hence, $H(u, \vec{x}) = \forall z \leq s(\vec{x})\ G(u, \vec{x}, z)$ where $G(u, \vec{x}, z)$ begins with a bounded existential quantifier and hence is in Σ^b_{k-1}. Since PV is a universal theory, by Theorem 3.10, there exist a uniform PV-reduction program P_u from $H(u+1, \vec{x})$ to $H(u, \vec{x})$ for $u < p(|\vec{x}|)$; a PV-reduction program N from $H(0, \vec{x})$ to $\forall y \leq t(x)\ \neg A(x, y)$ and finally a PV-reduction program K from $\bot$ to $H(p(|\vec{x}|), \vec{x})$. The idea is using the power to decide Σ^b_{k-1} formulas and in needed cases finding the witnesses for those decisions, to simplify the reduction programs. We will simplify the reduction program from $H(u+1, \vec{x})$ to $H(u, \vec{x})$, the cases for the other two are similar.

For simplicity, use z' for z in $H(u, \vec{x})$ to have $H(u, \vec{x}) = \forall z' \leq s(\vec{x})\ G(u, \vec{x}, z')$ and $H(u+1, \vec{x}) = \forall z \leq s(\vec{x})\ G(u+1, \vec{x}, z)$. Using the PV-reduction program P_u, we write an algorithm in $\Box^p_{k-1}$ to find z' from z. W.l.o.g we can assume that the program begins with reading z. The algorithm M_u is defined as the following: Begin with the sequent

$$S(P_u^{<1}) = \forall z' \leq s(\vec{x}) G(u, \vec{x}, z') \Rightarrow G(u+1, \vec{x}, z)$$

Check the truth value of $z \leq s(\vec{x}) \rightarrow G(u+1, \vec{x}, z)$. If it is true, halt and answer 0. If not, follow the program in a way that all added simpler formulas to the left hand-side (right hand-side) of $S(P_u^{<1})$ becomes true (false). More precisely, at the stage m of the program, if P_m is the instruction [**Read** $X \leq t(\vec{x})$] and if the formula $\forall X \leq t(\vec{x}) C(X)$ that is occurred in the right hand-side of $S(P^{<m})$ is false ($\exists X \leq t(\vec{x}) C$ in the left hand-side is true), find X such that $X \leq t(\vec{x})$ and $C(X)$ becomes false (true). If not, continue. For the instruction [**Compute** Y **by** $r \leq t(\vec{x})$], if $Y = z'^k$ for some k and $t(\vec{x}) = s(\vec{x})$, check if $G(u, \vec{x}, r)$ is true or false. If it is false then halt and answer r. If not, continue. If $Y \notin \{z'^k\}_{k \geq 0}$, then continue.

The algorithm definitely halts and finds $r(u, \vec{x}, z)$ such that both

$$z \leq s(\vec{x}) \rightarrow r(u, \vec{x}, z) \leq s(\vec{x})$$

and

$$z \leq s(\vec{x}) \rightarrow [G(u, \vec{x}, r(u, \vec{x}, z)) \rightarrow G(u+1, \vec{x}, z)]$$

become valid. The reason is simple. If the algorithm does not find such an r, it must reach the end of the program. Based on our construction, all added formulas to the right hand-side is false and all added formulas in the left hand-side is true. But there exists a quantifier-free subsequent of $S(P_u^{<l+1})$ such that $\mathrm{PV} \vdash S'$. Since S' consists of quantifier-free formulas, it should consist of added simpler formulas which implies that the left hand-side of S' is true while its right hand-side is false. Hence, S' is false. Therefore, the algorithm halts. But if it halts, there are two possibilities, either at the first stage $G(u+1, \vec{x}, z)$ is false, or in some stage there exists an r such that $G(u, \vec{x}, r(u, \vec{x}, z))$ is false. In both cases we have

$$z \leq s(\vec{x}) \rightarrow [G(u, \vec{x}, r(u, \vec{x}, z)) \rightarrow G(u+1, \vec{x}, z)].$$

We also have

$$z \leq s(\vec{x}) \rightarrow r(u, \vec{x}, z) \leq s(\vec{x})$$

In the first case because the output of the algorithm is 0 and in the second case, because we faced the instruction [**Compute** Y **by** $r \leq s(\vec{x})$] whose definition implies the claimed bound. Hence, the claim follows.

Now we show that the algorithm M_u computes a function in $\square_k^p$. Note that the algorithm begins with checking $z \leq s(\vec{x}) \rightarrow G(u+1, \vec{x}, z)$ which is in Σ_{k-1}^b. Then in each stage of the reduction program, if the instruction is [**Read** $X \leq t(\vec{x})$], the algorithm checks the truth value of an existential sub-formula of $\forall z' \leq s(\vec{x}) G(u, \vec{x}, z')$ or a universal sub-formula of $G(u+1, \vec{x}, z)$

which implies that the formula is in Σ^b_{k-1}. And finally if the instruction is [**Compute** Y **by** $r \leq t(x)$], at the worst case, we have to check $G(u, \vec{x}, r)$ which is also in Σ^b_{k-1}. Hence, the algorithm is a constant number of Σ^b_{k-1} oracle questions and thus is in $\square^p_k$.

Now let us investigate how complex this halting argument is. Since the length of the reduction program is a constant and

$$\mathrm{PV} \vdash \forall u < p(|\vec{x}|)\ H(u, \vec{x}) \rightarrow H(u+1, \vec{x})$$

it is easy to formalize the above mentioned argument in PV to show that

$$\forall u < p(|\vec{x}|) \forall z \leq s(\vec{x})[\exists w Com(\vec{x}, z, M_u, w) \rightarrow [G(u, \vec{x}, out(w)) \rightarrow G(u{+}1, \vec{x}, z)]] \quad (*)$$

and

$$\forall u < p(|\vec{x}|) \forall z \leq s(\vec{x})[\exists w Com(\vec{x}, z, M_u, w) \rightarrow out(w) \leq s(\vec{x})] \quad (**)$$

Now apply the same argument for the PV-reduction programs N and K to have:

(*i*) $\forall z \leq s(\vec{x})[\exists w Com(\vec{x}, z, N, w)) \rightarrow [(out(w) \leq t(\vec{x}) \rightarrow \neg A(\vec{x}, out(w))) \rightarrow G(0, \vec{x}, z)]]$.

(*ii*) $\exists w Com(\vec{x}, K, w) \rightarrow [G(p(|\vec{x}|), \vec{x}, out(w)) \rightarrow \bot]$.

(*iii*) $\exists w Com(\vec{x}, K, w) \rightarrow out(w) \leq s(\vec{x})$

Now define the algorithm M as running K on 0, then put M_u's end to end beginning from $u = p(|\vec{x}|)$ till $u = 0$ and at last run N. We claim that this M works. First note that M is a result of polynomially many computational steps in $\square^p_k$ and hence it is also in $\square^p_k$. Secondly note that by the length induction in PV on $p(|\vec{x}|) \dot{-} u$ and using $(*)$ and $(**)$ we can prove

$$\mathrm{PV} \vdash \exists w Com(\vec{x}, M, w) \rightarrow \forall u < p(|\vec{x}|)\ out(w_u) \leq s(\vec{x}).$$

and

$$\mathrm{PV} \vdash \exists w Com(\vec{x}, M, w) \rightarrow \forall u < p(|\vec{x}|) \neg G(u, \vec{x}, out(w_u)).$$

Hence

$$\mathrm{PV} \vdash \exists w Com(\vec{x}, M, w) \rightarrow [out(w) \leq t(\vec{x}) \wedge A(\vec{x}, out(w))]$$

which complete the proof. $\square$

References

[1] A. Akbar Tabatabai, Proof Mining in Bounded Arithmetic, preprint.

[2] A. Beckmann and S. R. Buss, Characterization of Definable Search Problems in Bounded Arithmetic via Proof Notations, Ontos Verlag, 2010, pp. 65-134.

[3] A. Beckmann and S. R. Buss, Polynomial local search in the polynomial hierarchy and witnessing in fragments of bounded arithmetic, Journal of Mathematical Logic, 9 (2009), pp. 103-138.

[4] S. R. Buss, Bounded Arithmetic, Bibliopolis, Naples, Italy, 1986.

[5] L. A. Kolodziejczyk, P. Nguyen, and N. Thapen, The provably total NP search problems of weak second-order bounded arithmetic, Annals of Pure and Applied Logic, 162 (2011).

[6] J. Krajicek, P. Pudlak, G. Takeuti, Bounded arithmetic and the polynomial hierarchy, Annals of Pure and Applied Logic, 52: 143-53.

[7] H. Schwichtenberg, A. Troelstra, Basic Proof Theory. No. 43 in Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, Cambridge, UK (1996).

[8] A. Skelley and N. Thapen, The provably total search problems of bounded arithmetic, Proceedings of the London Mathematical Society, 103 (2011), pp. 106-138.

[9] N. Thapen, Higher complexity search problems for bounded arithmetic and a formalized no-gap theorem, Archive for Mathematical Logic, Vol 50:7-8, pages 665-680, 2011.