

Proof Schemata for PA : An Approach to Constructing Conservative Reflection Principles

David M. Cerna*

Research Institute for Symbolic Computation
Johannes Kepler University
Linz, Austria
david.cerna@risc.jku.at

Anela Lolic†

Institute of Discrete Mathematics and Geometry
TU Wien
Vienna, Austria
anela@logic.at

Abstract

Induction is typically formalized as a rule or axiom extension of the **LK**-calculus. While this extension of the sequent calculus is simple and elegant, proof transformation and analysis can be quite difficult. Most alternative sequent calculus formalizations of induction do not extend *Herbrand's* theorem and are thus not fully adequate for proof analysis. In this work we extend the existing formalism of proof schemata, a recursive formulation of induction particularly suited for proof analysis, and show equivalence between **PA** and a fragment of our extended formalism. This relationship provides a conservative reflection principle between **PA** and an alternative proof formalism. A schematic approach need not be limited to arithmetic and should be investigated further.

Keywords Induction, Proof Schemata, Proof Analysis, Peano Arithmetic, Reflection Principles

1 Introduction

Proof schemata serve as an alternative proof formulation through primitive recursive proof specification. Essentially, the local soundness of a individual *proof component* is sacrificed, but globally the collection of components is sound. This property is illustrated in [11] where a calculus integrating this global soundness is introduced. The seminal work concerning “proof as schema” was focused on proof analysis of Fürstenberg’s proof of the infinitude of primes by Baaz et al. [3] using a rudimentary schematic formalism and **CERES** [4]. Schematic proof representation excels at proof analysis and transformation without “unrolling” the formal

proof, thus making it particularly suited for analysis of inductive reasoning. For example, Herbrand’s theorem can be extended to an expressive fragment (*k*-induction) of proof schemata [14, 19].

While *k*-induction lacks in expressive power when compared to theories such as **PA** [23], there is ample evidence showing that it is provability-wise quite expressive¹. However, the formalism is quantifier free over the numeric sort and severely restricts the proof structure, which is of interest to proof analysis and transformation. For example, the infinitary pigeonhole principle (IPP) [9, 10] can be elegantly formalized as a proof schema using Π_2 -cuts, though an important proof skeleton resulting from proof analysis, using atomic cuts only, is beyond the representational power of the current formal language. Most of the analysis had to be done outside the existing methodological framework [9]. Analysis of a restricted version of IPP [10] resulted in a proof skeleton expressible by the formal language presented in [14]. However, this restriction was designed to allow analysis in the existing framework by removing the complex combinatorics from which the full IPP statement is entailed (see [9]). Once again, the most interesting part of the proof had to be removed to fit the structural constraints of the formal system.

Other existing alternative formalisms for arithmetic [7, 8, 20] were developed without a concern for proof analysis and thus lack the construction of proof normal forms with *subformula-like properties*² and analytic tools applicable to proofs in a compressed state, i.e. prior to unrolling their recursive specification.

A recently developed proof analysis method [19] based on the schematic theorem proving work pioneered by V. Aravantinos et al. [1, 2] manages to preserve the analyticity, usually associated with the subformula property, in the presence of propositional cuts. While in some sense the method is cut-elimination complete in that it can provide the substitutions needed to construct a Herbrand sequent, it relies on

*This research is supported by the FWF project P28789-N32.

†This research is supported by the FWF project I-2671-N35.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

???, ???, ???

© 2018 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM. . \$15.00
<https://doi.org/10.1145/nnnnnnn.nnnnnnn>

¹See Gentzen [15], the *SiLK*-calculus Cerna & Lettmann [11], and Curry’s formalization of primitive recursive arithmetic [13].

²A proof fulfilling the subformula property can be referred to as *analytic*. By subformula-like, we mean that the proof is non-analytic, but still allows the extraction of objects important for proof analysis which require some form of analyticity.

a schematic superposition prover [2] which cannot deal with the complexities of most interesting k -inductive arguments. Even though, k -induction is quite expressive in theory, in practice we cannot expect the method of [19] to perform proof analysis on arbitrarily complex arithmetic statements. Also, the complexity of the formalism introduced in [19] makes the experimental approach taken in [10] quite a challenging endeavor, an approach which has shown promise [9]. Moreover, the formalism is based on a single parameter, i.e. nested induction is not allowed. Though this is not necessarily a weakness [11, 15], it does remove quantification over inductive arguments, the essential difference between **PA** and Primitive Recursive Arithmetic (**PRA**) [13].

In this work we provide a formalism which is provability-wise at least as expressive as **PA** without restricting the structure of the proof or the inductive argument. We do so by constructing proof schemata over a well-founded ordering and allowing multiple free parameters. Multi-parameter schemata provide quantifier introduction over numeric terms without the loss of properties essential for schematic proof analysis, and thus allowing us to formalize strong totality statements³. An essential proof analytic tool of [14], schematic characteristic clause set, can be generalized to our formalism and insofar support the experimental approach of [10]. Though, a corresponding schematic resolution calculus (as defined in [14]) has yet to be developed. We plan to address this in future work by applying the methods and techniques developed here to schematic resolution refutations. We expect such investigations to provide further insight into automated deduction for recursively defined formulas and clause sets and provide a theoretical foundation supporting investigation into interactive theorem proving methods for schematic resolution. Furthermore, proof analysis carried out in our formalism will in all likelihood provide interesting and complex examples (as it has done so far) fueling the research areas mentioned above.

Furthermore, proof schemata as presented in this work provide a formalism allowing induction over arbitrary function symbols, not just the successor. While we do not take advantage of this property in this paper it is a result of the recursive formalization presented which does not restrict the terms passed through *links* other than requiring them to be ordered. This is of most interest to the usage of reflection principles as in Parikh's results [6, 21] settling the Kreisel conjecture [5, 18] for the monadic version of **PA** *. However, in our case we present a reflection principle which is conservative over **PA**, which to the best of our knowledge, is the first of its kind. This approach, that is constructing a reflection principle by formulating a schematic version of a theory is not limited to arithmetic. We foresee application of similar methodology to other problems and see our

³As noted in [16], a single inductively introduced quantifier suffices for formalizing **PA**.

approach as a method of formulating logical relationships between a classical formulation of a theory and its schematic counterpart.

2 From Proof Schema to P -schema

In this section we introduce the previous formalization of the schematic language and proof schemata, introduce our new formalism of P -schema, and provide a comparison of the two formalisms.

We work in a setting with the sort ω representing the natural numbers, o representing bool. The language consists of countable sets of variables and sorted n -ary function and predicate symbols. We associate with every n -ary function f a tuple of sorts $(\tau_1, \dots, \tau_n, \tau)$ with the interpretation $f : \tau_1 \times \dots \times \tau_n \rightarrow \tau$, analogously we do the same for predicate symbols.

Terms are built from variables and function symbols in the usual way. We assume the constant function symbols $0 : \omega$ and $s : \omega \rightarrow \omega$ (zero and successor) to be present. Formulas are built as usual from atoms using the logical connectives $\neg, \wedge, \vee, \rightarrow, \forall, \exists$.

2.1 Proof Schemata and the Schematic Language

We now introduce *proof schemata* as presented in [11, 12, 14, 19], but without the first-order term sort ι . We are limited to predicate symbols with at most a single numeric index. Numeric terms are ground terms $\mathcal{G}_{\mathbb{N}}$ constructed from the alphabet $\{0, s(\cdot)\}$ with the addition of a free parameter denoted by n . Note that we will use lower-case Greek characters α, β, γ to represent ground numeric terms.

We work with a *schematic first-order language* allowing the specification of an (infinite) set of first-order formulas by a finite term. Therefore we allow *defined function symbols*, i.e. primitive recursively defined functions, in the language. Analogously, we use *defined predicate symbols* to build *formula schemata*, a generalization of formulas including defined predicate symbols defined inductively using the standard logical connectives from uninterpreted and defined predicate symbols. Defined symbols will be denoted by $\hat{\cdot}$, i.e. \hat{P} . We assume a set of convergent rewrite rules \mathcal{E} (equational theory) for defined function and predicate symbols. The rules of \mathcal{E} are of the form $\hat{f}(\bar{t}) = E$, where \bar{t} contains no defined symbols, and either \hat{f} is a function symbol and E is a term or \hat{f} is a predicate symbol and E is a formula schema. The rules can be applied in both directions, i.e. the \mathcal{E} -rule is reversible.

A *schematic sequent* is a pair of multisets of formula schemata Δ, Π denoted by $\Delta \vdash \Pi$. We will denote multisets of formula schemata by uppercase greek letters. We reuse this construction for P -schema.

Definition 2.1 (LKE). Let \mathcal{E} be an equational theory. **LKE** is an extension of **LK** by the \mathcal{E} inference rule

$$\frac{S(t)}{S(t')} \mathcal{E}$$

where the term or formula schema t in the sequent S is replaced by a term or formula schema t' for $\mathcal{E} \models t = t'$.

Let $S(\bar{x})$ be a sequent and \bar{x} a vector of free variables, then $S(\bar{t})$ denotes $S(\bar{x})$ where \bar{x} is replaced by \bar{t} , where \bar{t} is a vector of terms of appropriate type. $\frac{\varphi(a)}{S(a)}$, where $S(\bar{x})$ is a schematic sequent and φ some proof is called a *proof link rule*. The sequent calculus **LKS** consists of the rules of **LKE** extended by the proof link rule.

Proof links act as placeholders for proofs and during evaluation they are replaced by the proof corresponding to the proof symbol of the link. Proof schemata are a sequence of *proof schema components* with a restriction on the usage of proof links.

Definition 2.2 (Proof schema component). Let ψ be a proof symbol and k an expression with free variable n . A *proof schema component* C is a triple $(\psi, \pi, v(k))$ where π is an **LKS**-proof without proof links and $v(k)$ is an **LKS**-proof containing proof links. The end-sequents of the proofs are $S(0)$ and $S(k)$, respectively. Given a proof schema component $C = (\psi, \pi, v(k))$ we define $C.1 = \psi$, $C.2 = \pi$, and $C.3 = v(k)$.

Definition 2.3. Let C_1 and C_2 be proof schema components such that $C_{1.1}$ is distinct from $C_{2.1}$. We say $C_1 >^* C_2$ if there are no links from C_2 to C_1 and all links that call C_1 or C_2 are of the following form:

$$\frac{(C_{1.1}, k')}{S(k')} \quad \frac{(C_{2.1}, t)}{S'(t)}$$

for t s.t. $\mathcal{V}(t) \subseteq \{n\}$ and k' is a sub-term of k . $S(x)$ and $S'(x)$ are the end sequents of components C_1 and C_2 , respectively. Let Ψ be a set of proof schema components. We say $C_1 > C_2$ if $C_1 >^* C_2$ and $C_1 >^* D$ holds for all proof schema components D of Ψ with $C_2 >^* D$.

Definition 2.4 (Proof schema [14]). Let C_1, \dots, C_m be proof schema components s.t the $C_{i.1}$ are pairwise distinct. Let the end sequents of C_1 be $S(0)$ and $S(k)$. We define $\Psi = \langle C_1, \dots, C_m \rangle$ as a *proof schema* if $C_1 > \dots > C_m$. We call $S(k)$ the end sequent of Ψ .

Example 2.5 (Proof schema). Let us define a proof schema $\langle \langle \varphi, \pi, v(k) \rangle \rangle$ with end-sequent (schema)

$$P(0), \bigwedge_{i=0}^n (P(i) \rightarrow P(i+1)) \vdash P(n+1).$$

using the equational theory:

$$\mathcal{E} = \left\{ \begin{array}{l} \bigwedge_{i=0}^0 P(i) \rightarrow P(i+1) = P(0) \rightarrow P(1); \\ \bigwedge_{i=0}^{n+1} P(i) \rightarrow P(i+1) = \\ P(n+1) \rightarrow P(n+2) \wedge \bigwedge_{i=0}^n P(i) \rightarrow P(i+1) \end{array} \right\}.$$

π is as follows:

$$\frac{\frac{P(0) \vdash P(0) \quad P(1) \vdash P(1)}{P(0), P(0) \rightarrow P(1) \vdash P(1)} \rightarrow : l}{P(0), \bigwedge_{i=0}^0 P(i) \rightarrow P(i+1) \vdash P(1)} \mathcal{E}$$

$v(k)$ is as follows where $k = n + 1$:

$$\text{cut} \frac{\frac{\frac{(v_1(k)) \quad (v_2(k))}{S(v_1(k)) \quad S(v_2(k))}}{P(0), \bigwedge_{i=0}^n (P(i) \rightarrow P(i+1)), P(n+1) \rightarrow P(n+2) \vdash P(n+2)}}{\dots \wedge : l \text{ and } c : l \dots}}{P(0), \bigwedge_{i=0}^{n+1} (P(i) \rightarrow P(i+1)) \vdash P(n+2)}$$

where

$$S(v_1(k)) \equiv P(0), \bigwedge_{i=0}^n (P(i) \rightarrow P(i+1)) \vdash P(n+1),$$

$$S(v_2(k)) \equiv P(n+1), P(n+1) \rightarrow P(n+2) \vdash P(n+2),$$

$v_1(k)$ is

$$\frac{\varphi(n)}{P(0), \bigwedge_{i=0}^n (P(i) \rightarrow P(i+1)) \vdash P(n+1)}$$

and $v_2(k)$ is

$$\frac{P(n+1) \vdash P(n+1) \quad P(n+2) \vdash P(n+2)}{P(n+1), P(n+1) \rightarrow P(n+2) \vdash P(n+2)}$$

2.2 The P-schema Formalism

We extend the formalism defined in the previous section to construct **P-schemata**. Other than the ground numeric terms we also have three types of parameters: *active* parameters \mathcal{N}_a , *passive* parameters \mathcal{N}_p , and *internal* parameters \mathcal{N}_i ⁴. We denote the active parameters by lower-case Latin characters n, m, k , passive parameters by lower-case bold Greek characters α, β, γ , and internal parameters by lower-case bold Latin characters $\mathbf{n}, \mathbf{m}, \mathbf{k}$. We define the set of numeric terms containing these parameters as $\mathcal{A}_{\mathbb{N}}, \mathcal{P}_{\mathbb{N}}, \mathcal{I}_{\mathbb{N}}$, respectively. In general, the set of schematic terms will be denoted by $\mathcal{S}_{\mathbb{N}} = \mathcal{A}_{\mathbb{N}} \cup \mathcal{P}_{\mathbb{N}} \cup \mathcal{G}_{\mathbb{N}} \cup \mathcal{I}_{\mathbb{N}}$. To simplify reading, we will denote the successor of an active parameter n by n' rather than $s(n)$.

Concerning quantification, one ought to consider passive parameters as *eigenvariables*. Both active and internal parameters as well as terms which contain them cannot be quantified. These parameters play a computational role and thus quantification of these symbols can lead to unsound derivation.

We will be mainly concerned with schematic sequents S referred to as (n, \mathcal{I}) -sequent, where n is an active parameter and \mathcal{I} is a set of internal parameters, where the free active and internal parameters of S are n and \mathcal{I} , respectively. A sequent without active parameters will be referred to as an \mathcal{I} -sequent. We do not consider sequents with more than one active parameter. Our primitive recursive formalization requires pairing the end sequent of a basecase proof (an \mathcal{I} -sequent) with the end sequent of a stepcase proof (an (n, \mathcal{I}) -sequent); we call these pairs of sequents *inductive pairs*. For example, given an (n, \mathcal{I}) -sequent S an inductive pair based on S would be $(S, S\{n \leftarrow \alpha\})$ where $\alpha \in \mathcal{G}_{\mathbb{N}}$.

⁴A related terminology can be found in [22] which discusses a construction similar to ours.

We consider an extension of the **LKE**-calculus [14] over the extended term language defined above, which we refer to as the *multivariate LKE*-calculus (*mvLKE*-calculus). By multivariate we are referring to the three types of variable symbols. Before moving on to the definition of **P**-schemata we further distinguish the three types of parameters. By active parameter we are referring to the parameter over which the induction is performed. Every *mvLKE*-proof has at most one distinct active parameter. Passive parameters are used to mark already occurring inductions, or as mentioned before, act as eigenvariables. There is no limit to the number of passive parameters in a given proof. Internal parameters are used to pass information through the “links” which we will discuss shortly. Thus, an *mvLKE*-derivation can end with a schematic sequent containing internal parameters, but an *mvLKE*-proof ends with a schematic sequent without internal or active parameters. The end-sequent of an *mvLKE*-proof (*mvLKE*-derivation) φ will be denoted as $es(\varphi)$ and the set $\mathcal{V}_x(S)$ for $x \in \{a, p, i\}$ will denote the active, passive and internal parameters occurring in the sequent S , respectively. Notice that the calculus introduced so far cannot construct *mvLKE*-proofs unless the entire derivation is active and internal parameter free. To deal with this issue we introduce the concept of schematic proof (derivation) for the *mvLKE*-calculus.

As in section 2.1, we define schematic proof (derivation) as a finite set of *components* which can “link” together using *links* defined analogously to proof links. Note any type of parameter is allowed in the arguments of the links. Furthermore, we assume a countably infinite set \mathcal{B}^5 of *proof symbols* denoted by $\varphi, \psi, \varphi_i, \psi_j$.

Definition 2.6 (*mvLKS*). The *mvLKS*-calculus is an extension of *mvLKE*, where links may appear at the leaves of a proof (derivation).

Note that *mvLKS*-proofs do not have to be valid by construction and requires an external soundness condition. This soundness condition is provided by the **P**-schema construction. Also, we will refer to an *mvLKS*-derivation as *inactive* if it does not contain an active parameter and $\{n\}$ -active if it contains only the active parameter n .

Example 2.7. Consider the following \mathcal{E} theory

$$\mathcal{E} = \{\widehat{a}(s(n), \beta) = s(\widehat{a}(n, \beta)); \widehat{a}(0, \beta) = \beta\}.$$

Let $\pi =$

$$\frac{\frac{\vdash 0 = 0}{\vdash a(0, 0) = 0} \mathcal{E}}{\vdash a(0, 0) = a(0, 0)} \mathcal{E}$$

and let $\nu =$

$$\frac{\begin{array}{c} \dots\dots\dots \chi(n) \dots\dots\dots \\ \vdash a(n, 0) = a(0, n) \end{array} \quad \frac{S_1(\nu_1)}{\vdash a(n', 0) = a(0, n')} \quad \text{cut}}{\vdash a(n', 0) = a(0, n')}$$

⁵Think of the German word *Beweis* meaning proof.

where $S_1(\nu_1) \equiv a(n, 0) = a(0, n) \vdash s(a(n, 0)) = s(a(0, n))$. π is an *mvLKS*-proof and ν is an *mvLKS*-derivation. Also, ν contains a link to the proof symbol χ and is $\{n\}$ -active. Note that the end-sequent of ν , $es(\nu)$ is $es(\chi(n))\{n \leftarrow n'\}$. Moreover, $es(\nu)$ is an (n, \emptyset) -sequent, i.e. internal parameter free. Its inductive pair also contains the sequent $es(\nu)\{n \leftarrow 0\} \vdash a(0, 0) = a(0, 0)$, the end-sequent of π (an \emptyset -sequent). The triple (χ, π_1, ν_1) is referred to as an (n, \emptyset) -component.

Definition 2.8 ((n, \mathcal{I}) -component). Let $\psi \in \mathcal{B}$, $n \in \mathcal{N}_a$ and $\mathcal{I} \subset \mathcal{N}_i$. An (n, \mathcal{I}) -component C is a triple (ψ, π, ν) where π is an inactive *mvLKS*-derivation ending with $S\{n \leftarrow \alpha\}$ and ν is an $\{n\}$ -active *mvLKS*-derivation ending in an (n, \mathcal{I}) -sequent S , the inductive pair of $S\{n \leftarrow 0\}$. Given a component $C = (\psi, \pi, \nu)$ we define $C.1 = \psi$, $C.2 = \pi$, and $C.3 = \nu$. We refer to $es(C) = S$ as the *end sequent* of the component C .

When the extra terms are not necessary for understanding, we will refer to an (n, \mathcal{I}) -component as a component.

A schematic proof is defined over finitely many components, that are connected by links. This means, that a component may contain links to other components. So far, there is no restriction on the usage of links, which must obey some conditions in order to preserve soundness. In fact, whenever a component C contains a link to another component D , it has to be ensured that the passive parameters occurring in $es(D)$ occur in the sequent associated with the link in C as well. This condition is formalized in the following definition.

Definition 2.9 (Linkability). Two components C and D are said to be (C, D) -linkable if for each sequent S in C that is associated with a proof link to D it holds that $\mathcal{V}_p(es(D)) \subseteq \mathcal{V}_p(S)$. We say they are *strictly* (C, D) -linkable if it holds that $\mathcal{V}_p(es(D)) \subseteq \mathcal{V}_p(es(C))$.

The restriction on linkable components is used to define an ordering on the components occurring in a schematic proof.

Definition 2.10 (Linkability ordering). Let C_1 and C_2 be distinct components such that they are (C_1, C_2) -linkable. Then we say that $C_1 < C_2$. If C_1 and C_2 are strictly (C_1, C_2) -linkable, we say that $C_1 <_s C_2$.

When constructing recursive proofs it is essential to avoid mutual recursion and infinite cycles. The restriction on the number of active parameters per sequent deals with mutual recursion and the linkability ordering provides a method to deal with infinite cycles⁶. However, the linkability ordering alone does not prevent infinite cycles; it is defined over the set of all components and allows the definition of mutually linkable components, i.e. (C_1, C_2) -linkable and (C_2, C_1) -linkable components C_1, C_2 . To avoid this problem we define **P**-schemata over a subordering of the linkability ordering which is well founded. In this work we restrict this

⁶We do not consider proof by infinite descent [8].

where $S(v_{3_1}) \equiv \vdash a(\alpha, a(1, n)) = a(a(\alpha, 1), n)$ and $v_{3_1} =$

$$\frac{\dots\dots\dots \varphi(\alpha, 1, n) \dots\dots\dots}{\vdash a(\alpha, a(1, n)) = a(a(\alpha, 1), n)}$$

Notice that ξ is the least element of the order $<$ and the following relations concerning $<$ are also defined: $\xi < \varphi$, $\xi < \psi$, $\psi < \chi$. Also, once again Φ' defines a *mvLKS*-derivation rather than a proof, however, we can perform a similar extension as before to construct a proof. Furthermore we can quantify the passive parameters of the schema and derive the precise statement of commutativity as one would derive in **PA**.

We can formalize the construction of an *mvLKS*-proof from an *mvLKS*-derivation as follows:

Theorem 2.15. *Let $\Phi = \langle C_1, \dots, C_\alpha \rangle$ be a **P**-schema resulting in an *mvLKS*-derivation such that $0 < |\mathcal{V}_i(\text{es}(\Phi))| + |\mathcal{V}_a(\text{es}(\Phi))|$ and $|\mathcal{V}_p(\text{es}(\Phi))| + |\mathcal{V}_i(\text{es}(\Phi))| + |\mathcal{V}_a(\text{es}(\Phi))| \leq |P|$. Then there exists $\Phi' = \langle C, C_1, \dots, C_\alpha \rangle$ resulting in an *mvLKS*-proof.*

Proof. Essentially the translation of Example 2.13. \square

We will refer to such proof schemata as *complete* if every sub **P**-schema of Φ is also non-computational.

In the following sections we show that the results and concepts of [19] can be extended to our more general formalization.

2.3 Evaluating P-Schemata

Like the proof schema of previous work, **P**-schema represents infinite sequences of proofs. We extend the soundness result of proof schema to the **P**-schema case. However, the evaluation procedure is a bit more involved given the more complex schematic language of **P**-schemata.

Definition 2.16 (Evaluation of **P**-schemata). Let Φ be a complete **P**-schema $\langle C_1, \dots, C_\alpha \rangle$. We define the rewrite rules for links

$$\hat{\psi}_i(0, \mathcal{I}) \rightarrow \pi \quad \hat{\psi}_i(s(n), \mathcal{I}) \rightarrow v$$

where $\hat{\psi}_i$ corresponds to C_i with active parameter n . The rewrite system for links is the union of these rules. Moreover, for a substitution $\sigma : \mathcal{N}_p \rightarrow \mathcal{G}_{\mathbb{N}}$ with domain P , we define $\Phi\sigma = \hat{\psi}_1\sigma$ as the normal form of Φ under these rewrite rules and \mathcal{E} .

Lemma 2.17. *Let Φ be a complete **P**-schema and $\sigma : \mathcal{N}_p \rightarrow \mathcal{G}_{\mathbb{N}}$ with domain P a substitution. The rewrite system for the links of Φ is strongly normalizing and confluent, s.t. $\Phi\sigma$ is an **LK**-proof.*

Proof. By the restriction on occurrences of links, a proof schema can be seen as a set of primitive recursive definitions, and the rewrite rules for links are the standard rules for these definitions. It is well-known that such rewrite systems are strongly normalizing, see [17]. Finally, by the restriction to complete **P**-schema links will not occur in the

normal form and $\Phi\sigma$ is an *mvLKE*-proof. Furthermore, since all \mathcal{E} -inferences in this proof are trivial and there are no parameters, we may consider it as an **LK**-proof. \square

The constraints of Lemma 2.17 can be relaxed by allowing computational sub **P** schema. This can be handled by structural induction on the **P**-schema construction by showing that any complete **P**-schema could be unrolled into an **LK**-proof and therefore is soundly constructed. Also, showing that a **P**-schema containing computation sub **P**-schemata, treating any link to a sub **P**-schema as an axiom, unrolls into an **LK**-proof (modulo a theory extension) and is therefore soundly constructed.

In light of these details the soundness of **P**-schemata w.r.t. *mvLKE* can be stated as follows:

Theorem 2.18 (Soundness of **P**-schemata). *Let Ψ be a **P**-schema and let $\sigma : \mathcal{N}_p \rightarrow \mathcal{G}_{\mathbb{N}}$ be a substitution. Then $\Psi\sigma$ is an **LK**-proof over a theory T of $\text{es}(\Psi)\sigma$, where T is the set of end-sequents associated with the computational sub **P**-schema Ψ .*

An interesting side note is that the calculus presented in [11] can easily integrate the formalization presented here and thus can provide a calculus for **P**-schema. We will consider this in future work.

Note that when measuring the size of **P**-schema, computational sub **P**-schema of the **P**-schema do not contribute to the measurement. This will be important for the definition of the *numeric P*-schema introduced in Section 5.

3 Local Induction and mvLKE

In [19], it was shown that proof schemata are equivalent to a particular fragment of arithmetic, i.e. the so called *k-simple induction*, which limits the number of inductive eigenvariables⁷ to one. The induction rule is as follows

$$\frac{F(k), \Gamma \vdash \Delta, F(s(k))}{F(0), \Gamma \vdash \Delta, F(t)} \text{IND}$$

where t is a term of the numeric sort such that t either contains k (k is a free parameter in the sense of [19]) or is ground. Adding the above rule to **LKE** resulted in the **LKIE**-calculus. We will refer to this calculus as the *simple LKIE*-calculus. The *mvLKE*-calculus and **P**-schema are related to a much more expressive induction rule. Essentially any term t can replace the active parameter n of the auxiliary sequent (including a term containing n) and the internal parameters can be instantiated with arbitrary terms. The instantiations must obey the restriction of at most one active parameter per schematic sequent. We refer to the calculus with the following induction rule as the *mvLKIE*-calculus:

$$\frac{F(n, \mathbf{m}_1, \dots, \mathbf{m}_\alpha), \Gamma \vdash \Delta, F(n', \mathbf{m}_1, \dots, \mathbf{m}_\alpha)}{F(0, \mathbf{a}_1, \dots, \mathbf{a}_\alpha), \Gamma \vdash \Delta, F(t, \mathbf{a}_1, \dots, \mathbf{a}_\alpha)} \text{mvIND}$$

⁷Inductive eigenvariables are eigenvariables occurring in the context of an induction inference rule.

where the \mathbf{m}_i are internal parameters which can be replaced by any schematic term, that is active, passive, ground, or another internal parameter.

We consider an $mv\mathbf{LKIE}$ -derivation ψ as an $mv\mathbf{LKIE}$ -proof if the end-sequent of ψ only contains passive parameters. We will first consider strict $mv\mathbf{LKIE}$ -proofs which, like strict links, require preservation of the passive parameters, i.e. all passive parameters used in the proof must show up in the end sequent.

Example 3.1. Here we present a strict $mv\mathbf{LKIE}$ -proof of the $\{\alpha, \beta\}$ -schema of Example 2.14. Notice how the links are replaced by the induction rules in a similar fashion to the k -induction conversion introduced in [19].

$$\begin{array}{c}
\vdots \\
\frac{a(n, a(\mathbf{m}, k)) = a(a(n, \mathbf{m}), k) \vdash \\ a(n', a(\mathbf{m}, k)) = a(a(n', \mathbf{m}), k)}{a(0, a(1, n)) = a(a(0, 1), n) \vdash \\ a(\alpha, a(1, n)) = a(a(\alpha, 1), n)} \quad mv\text{IND} \\
(v_1) \quad \frac{\vdots \\ \vdash a(\alpha, a(1, n)) = a(a(\alpha, 1), n)}{a(\alpha, n) = a(n, \alpha) \vdash \\ a(\alpha, a(1, n)) = a(n', \alpha)} \quad \text{cut} \\
\frac{a(\alpha, n) = a(n, \alpha) \vdash \\ a(\alpha, a(1, n)) = a(n', \alpha)}{a(\alpha, n) = a(n, \alpha) \vdash \\ a(\alpha, s(a(0, n))) = a(n', \alpha)} \quad \mathcal{E} \\
\frac{a(\alpha, n) = a(n, \alpha) \vdash \\ a(\alpha, n') = a(n', \alpha)}{a(\alpha, 0) = a(0, \alpha) \vdash \\ a(\alpha, \beta) = a(\beta, \alpha)} \quad mv\text{IND}
\end{array}$$

where v_1 is

$$\begin{array}{c}
\frac{a(n, 1) = a(1, n) \vdash \\ a(n', 1) = a(1, n') \vdash \\ \dots \\ a(0, 1) = a(1, 0) \vdash \\ a(\alpha, 1) = a(1, \alpha)}{\vdots} \quad mv\text{IND} \\
\frac{a(\alpha, n) = a(n, \alpha) \vdash \\ a(\alpha, n) = a(n, \alpha)}{\vdots} \quad \text{cut}
\end{array}$$

Note that the proof of equivalence between \mathbf{LKIE} -proofs and proof schemata provided in [19] does not directly use the k -simple induction restriction. What is important for the argumentation is that \mathbf{LKIE} -proofs are structured in similar fashion as proof schemata. Thus we can, for the most part, use the same arguments to prove the feasibility of translation for \mathbf{P} -schema and mv induction. This argument is easier to make when we enforce the proofs to be strict, i.e. transforming them into complete \mathbf{P} -schema, however such construction can be easily generalized to complete \mathbf{P} -schema by considering strict \mathbf{P} -schema as a base case for well-founded structural induction.

Lemma 3.2. *Let Ψ be a strict⁸ \mathbf{P} -schema with end-sequent \mathcal{S} . Then there exists a strict $mv\mathbf{LKIE}$ -derivation of \mathcal{S} .*

Proof. We can start by considering the proof of Proposition 3.13 from [19]. We know that proof schemata are $\{\alpha\}$ -schemata and only contain (n, I) -components. Thus, Proposition 3.13 from [19] provides a base case for the translation

⁸ $\mathcal{V}_p(es(\Psi)) \equiv P$

of strict \mathbf{P} -schema to $mv\mathbf{LKIE}$ -derivations. Now let us consider the case when we have more than one active parameter. This does not influence the construction outlined in [19] because there is only one active parameter per component. The only difference is that the translation of [19] must be applied to the base cases of the components as well. \square

Note that the translation defined in Lemma 3.2, along with Lemma 3.3, almost provides equivalence between strict \mathbf{P} -schema and primitive recursive arithmetic (**PRA**), though it is not clear if the \mathcal{E} rule provides a more expressive language than **PRA**. However, Lemma 3.3 clearly shows that **PRA** \subseteq strict \mathbf{P} -schema.

Lemma 3.3. *Let Π be a strict $mv\mathbf{LKIE}$ -derivation of \mathcal{S} containing α induction inferences of the form*

$$\frac{F_\beta(n, \mathbf{m}_1, \dots, \mathbf{m}_\gamma), \Gamma_\beta \vdash \Delta_\beta, F_\beta(n', \mathbf{m}_1, \dots, \mathbf{m}_\gamma)}{F_\beta(0, \mathbf{a}_1, \dots, \mathbf{a}_\gamma), \Gamma_\beta \vdash \Delta_\beta, F_\beta(t, \mathbf{a}_1, \dots, \mathbf{a}_\gamma)} \quad mv\text{IND}$$

where $1 \leq \beta \leq \alpha$, and if $\eta < \beta$ then the induction inference with conclusion

$$F_\beta(0, \mathbf{a}_1, \dots, \mathbf{a}_\gamma), \Gamma_\beta \vdash \Delta_\beta, F_\beta(t, \mathbf{a}_1, \dots, \mathbf{a}_\gamma)$$

is above the induction inference with conclusion

$$F_\eta(0, \mathbf{a}'_1, \dots, \mathbf{a}'_{\gamma*}), \Gamma_\eta \vdash \Delta_\eta, F_\eta(t, \mathbf{a}'_1, \dots, \mathbf{a}'_{\gamma*})$$

in Π . Then there exists a strict \mathbf{P} -schema with end-sequent \mathcal{S} .

Proof. Let T be the transformation taking an $mv\mathbf{LKIE}$ -derivation φ to an $mv\mathbf{LKS}$ -derivation by replacing the induction inferences $F_\eta(0, \mathbf{a}_1, \dots, \mathbf{a}_\gamma), \Gamma_\eta \vdash \Delta_\eta, F_\eta(t, \mathbf{a}_1, \dots, \mathbf{a}'_{\gamma'})$, $\eta < \beta$, with a proof link $\psi_\eta(t, \mathbf{a}'_1, \dots, \mathbf{a}'_{\gamma'})$. If the transformation reaches the induction inference β it replaces the $F_\beta(0, \mathbf{a}_1, \dots, \mathbf{a}_\gamma), \Gamma_\beta \vdash \Delta_\beta, F_\beta(t, \mathbf{a}_1, \dots, \mathbf{a}_\gamma)$ with a proof link $\psi_\beta(t, \mathbf{m}_1, \dots, \mathbf{m}_\gamma)$ and sequent of the proof link $F_\beta(0, \mathbf{m}_1, \dots, \mathbf{m}_\gamma), \Gamma_\beta \vdash \Delta_\beta, F_\beta(n, \mathbf{m}_1, \dots, \mathbf{m}_\gamma)$. The instantiation is placed in the construction of the component for the predecessor of β . Such a transformation obviously constructs a strict \mathbf{P} -schema from a strict $mv\mathbf{LKIE}$ -derivation.

We will inductively construct a strict \mathbf{P} -schema $\langle C_1, \dots, C_\alpha \rangle$ where $C_\beta = (\psi_\beta, \pi, \nu)$ has the end sequent $F_\beta(0, \mathbf{m}_1, \dots, \mathbf{m}_\gamma), \Gamma_\beta \vdash \Delta_\beta, F_\beta(n, \mathbf{m}_1, \dots, \mathbf{m}_\gamma)$ for some active parameter n . Assume that we have already constructed such proofs for $C_{\beta+1}, \dots, C_\alpha$ and consider the induction inference with the following main sequent $F_\beta(0, \mathbf{a}_1, \dots, \mathbf{a}_\gamma), \Gamma_\beta \vdash \Delta_\beta, F_\beta(t, \mathbf{a}_1, \dots, \mathbf{a}_\gamma)$. Let ξ be the derivation above the induction. We set π to $F_\beta(0, \mathbf{m}_1, \dots, \mathbf{m}_\gamma), \Gamma_\beta \vdash \Delta_\beta, F_\beta(0, \mathbf{m}_1, \dots, \mathbf{m}_\gamma)$ which by definition fulfills the requirements of links. Furthermore, let ν be the proof

$$\frac{\frac{\psi_\beta(n, \bar{\mathbf{m}}_\gamma) \quad T(\xi)}{S(\psi_\beta(n, \bar{\mathbf{m}}_\gamma)) \quad S(T(\xi))} \quad mv\text{IND}}{F_\beta(0, \bar{\mathbf{m}}_\gamma), \Gamma_\beta, \Gamma_\beta \vdash \Delta_\beta, \Delta_\beta, F_\beta(n', \bar{\mathbf{m}}_\gamma)} \quad c^* \\
F_\beta(0, \bar{\mathbf{m}}_\gamma), \Gamma_\beta \vdash \Delta_\beta, F_\beta(n', \bar{\mathbf{m}}_\gamma)$$

where $S(\psi_\beta(n, \bar{\mathbf{m}}_\gamma)) \equiv F_\beta(0, \bar{\mathbf{m}}_\gamma), \Gamma_\beta \vdash \Delta_\beta, F_\beta(n, \bar{\mathbf{m}}_\gamma), S(T(\xi)) \equiv F_\beta(n, \bar{\mathbf{m}}_\gamma), \Gamma_\beta \vdash \Delta_\beta, F_\beta(n', \bar{\mathbf{m}}_\gamma)$, which also clearly satisfies the requirement on links. Summarizing, C_β is a component with end-sequent $F_\beta(0, \bar{\mathbf{m}}_\gamma), \Gamma_\beta \vdash \Delta_\beta, F_\beta(n, \bar{\mathbf{m}}_\gamma)$. Linkability and the partial ordering come for free from the construction of strict *mvLKIE*-derivations. \square

Notice that Lemma 3.3 does not put a restriction on the number of passive parameters in the end sequent, but limits the partial ordering of components to a total linear ordering. A simple corollary of Lemma 3.3 removes the restriction on the ordering of components. Notice that proving the corollary requires the same induction argument over a more complex order structure (the linkability ordering). Essentially, we would have to join chains of components together using cuts.

Corollary 3.4. *Let Π be a strict *mvLKIE*-derivation of S . Then there exists a strict **P**-schema with end-sequent S .*

Concerning strict *mvLKIE*-derivations, notice that the need for passive, internal, and active parameters is no longer there. The three parameters aided the formalization of **P**-schema by removing mutual recursion and parameter instantiation, which are difficult to handle. Essentially, a reasonable class of strict **P**-schema could not be constructed without the three types of parameters. But for strict *mvLKIE*-derivations, the construction is obvious and enforced by the proof structure, thus, we can replace internal and active parameters by the corresponding constants and passive parameters. The resulting rule is essentially the induction rule of arithmetic. However, along with the \mathcal{E} rule, the language is at least a conservative extension of **PRA**. We show that for a particular choice of equational theory and using the standard equational axioms, the \mathcal{E} rule is admissible and thus strict *mvLKIE*-derivations are at least as expressive as **PRA** and by transitivity so is the strict **P**-schema formulation.

Furthermore, by dropping the strictness requirement, that is allowing computational sub **P**-schema, we can show that the **P**-schema formulation is at least as expressive as **PA**. As we mention above, this can be done by using the results of this section as a basecase and performing a structural induction over the construction of a **P**-schema.

4 **P**-schema, **PRA**, and **PA**

We will consider the **P**-schema formulation over the following equational theory

$$\mathcal{E}_{\text{PA}} = \left\{ \begin{array}{l} \widehat{a}(s(n), \beta) = s(\widehat{a}(n, \beta)); \widehat{a}(0, \beta) = \beta \\ \widehat{m}(s(n), \beta) = \widehat{a}(\widehat{m}(n, \beta), \beta); \widehat{m}(0, \beta) = 0 \end{array} \right\}$$

using the axioms of the formalization of Peano arithmetic and equational theory found in [23]. We will refer to these axioms as Ax_{PA} . We only consider a single two place propositional symbol which we will refer to as equality. We will refer to the calculus defined in [23] for **PA** as the **PA**-calculus.

Lemma 4.1. *Let Π be a strict *mvLKIE*-proof using Ax_{PA} and \mathcal{E}_{PA} . Then there exists a strict *mvLKIE*-proof Π' without the \mathcal{E} inference rule (Π' is \mathcal{E} -free).*

Proof. The rewrite rules of \mathcal{E}_{PA} are precisely the axioms of Ax_{PA} for addition and multiplication. Thus, from those axioms and the equational theory found in [23] anything provable by the \mathcal{E} inference rule can be proven using the above mentioned axioms and atomic cuts. \square

Now that we have \mathcal{E} -free strict *mvLKIE*-proofs we can consider translation to the **PA**-calculus without quantification. As the end sequent of a strict *mvLKIE*-proof only has passive variables. We can push the passive variables up the proof tree and replace each active variable by a fresh passive variable. Thus, the resulting proof only contains passive variables and constants and is a proof of the **PRA**-calculus, being that we have so far avoided quantification. Of course, a back translation can be performed, however doing so would not result in the same proof as the one we started with.

Theorem 4.2. *There exists a \mathcal{E} -free strict *mvLKIE*-proof of a sequent S iff there exists a **PRA**-calculus proof of S .*

Proof. As a consequence of pushing the passive parameters up the proof to the leaves we convert *mvLKIE* induction rules to standard induction rules, and thus **PRA**-calculus proofs. Furthermore a back translation is possible by reversing the method. \square

Example 4.3. The following proof of commutativity is the result of applying the translation from \mathcal{E} -free strict *mvLKIE*-proofs to **PA**-calculus proofs to the example first illustrated in Example 2.14.

$$\frac{\frac{\frac{\vdots}{a(\mathbf{v}, a(1, \boldsymbol{\gamma})) = a(a(\mathbf{v}, 1), \boldsymbol{\gamma}) \vdash} a(s(\mathbf{v}), a(1, \boldsymbol{\gamma})) = a(a(s(\mathbf{v}), 1), \boldsymbol{\gamma})}{a(0, a(1, \boldsymbol{\gamma})) = a(a(0, 1), \boldsymbol{\gamma}) \vdash} \text{IND}}{a(\boldsymbol{\alpha}, a(1, \boldsymbol{\gamma})) = a(a(\boldsymbol{\alpha}, 1), \boldsymbol{\gamma})} \text{IND}}{\frac{\vdots}{\vdash a(\boldsymbol{\alpha}, a(1, \boldsymbol{\gamma})) = a(a(\boldsymbol{\alpha}, 1), \boldsymbol{\gamma})} (v_1)} \text{cut} \frac{a(\boldsymbol{\alpha}, \boldsymbol{\gamma}) = a(\boldsymbol{\gamma}, \boldsymbol{\alpha}) \vdash}{a(\boldsymbol{\alpha}, a(1, \boldsymbol{\gamma})) = a(s(\boldsymbol{\gamma}), \boldsymbol{\alpha})} (1)$$

where v_1 is

$$\frac{\frac{\frac{\frac{\vdots}{a(\boldsymbol{\mu}, 1) = a(1, \boldsymbol{\mu}) \vdash} a(s(\boldsymbol{\mu}), 1) = a(1, s(\boldsymbol{\mu}))}{\vdots} \text{IND}}{a(0, 1) = a(1, 0) \vdash} a(\boldsymbol{\alpha}, 1) = a(1, \boldsymbol{\alpha})}{\vdots} \text{cut} \frac{a(\boldsymbol{\alpha}, \boldsymbol{\gamma}) = a(\boldsymbol{\gamma}, \boldsymbol{\alpha}) \vdash}{a(\boldsymbol{\alpha}, \boldsymbol{\gamma}) = a(\boldsymbol{\gamma}, \boldsymbol{\alpha})} \text{cut}$$

$$\begin{array}{c}
 \vdots \\
 (v_2) \quad \frac{a(\alpha, s(a(0, \gamma))) = a(s(\gamma), \alpha), \quad \alpha = \alpha, s(\gamma) = s(\gamma), \quad s(a(0, \gamma)) = s(\gamma) \vdash \quad a(\alpha, s(\gamma)) = a(s(\gamma), \alpha)}{\text{cut}} \\
 \vdots \\
 \frac{\frac{a(\alpha, \gamma) = a(\gamma, \alpha) \vdash \quad a(\alpha, s(\gamma)) = a(s(\gamma), \alpha)}{a(\alpha, 0) = a(0, \alpha) \vdash \quad a(\alpha, \beta) = a(\beta, \alpha)} \text{IND}}{\text{cut}}
 \end{array}$$

where v_2 is

$$\begin{array}{c}
 \frac{\frac{(1) \quad a(\alpha, \gamma) = a(\gamma, \alpha) \vdash \quad a(\alpha, a(1, \gamma)) = a(s(\gamma), \alpha), \quad \alpha = \alpha, s(\gamma) = s(\gamma), \quad a(1, \gamma) = s(a(0, \gamma)) \vdash \quad a(\alpha, s(a(0, \gamma))) = a(s(\gamma), \alpha)}{\text{cut}}}{\vdots} \\
 \frac{a(\alpha, \gamma) = a(\gamma, \alpha) \vdash \quad a(\alpha, s(a(0, \gamma))) = a(s(\gamma), \alpha)}{\text{cut}}
 \end{array}$$

To get from \mathbf{PRA} to \mathbf{PA} we need to remove the requirement of only considering strict $mv\mathbf{LKIE}$ -proofs. In terms of P -schema, this would mean allowing computational sub P -schema. We can build a $mv\mathbf{LKIE}$ -proof χ containing a sub-derivation ψ which is a strict $mv\mathbf{LKIE}$ -proofs by allowing strong quantification on the passive parameters of ψ in χ . The problem is that doing so can possibly destroy the translation of Section 3. To show this is not possible we just have to consider translation of χ in parts, first we translate ψ and then we translate χ without ψ , that is replacing ψ with a theory axiom when translating. Once we finish the translation of both parts we glue them back together to get a translation of the original proof χ . This argument can be formalized as mentioned in previous sections by performing a structural induction based on the above argument. This results in the following theorems:

Theorem 4.4. *There is a \mathbf{P} -schema Ψ with end-sequent S iff there is an $mv\mathbf{LKIE}$ -derivation of S .*

Proof. We can convert Ψ into an $mv\mathbf{LKIE}$ -derivation by structural induction over the number of passive parameters not associated with computational sub \mathbf{P} -schema. First considering strict \mathbf{P} -schema (Theorem 4.2) and then considering as the inductive hypothesis that the theorem holds for the first n computational sub \mathbf{P} -schema of Ψ and then showing it holds for $n + 1$. \square

Finally, we can extend the results of this section to \mathbf{PA} .

Theorem 4.5. *There is an \mathcal{E} -free $mv\mathbf{LKIE}$ -derivation of a sequent S iff there is a \mathbf{PA} -calculus proof of S .*

Proof. Note that the \mathbf{PA} induction rule is a special case of the $mv\mathbf{LKIE}$ induction rule and thus making backwards translation possible \square

5 A Conservative Reflection Principle

In the previous section we have shown that the induction principle of \mathbf{PA} can be understood as a recursive construction encapsulating the equational \mathbf{LK} -calculus. This translation provides a proof form for \mathbf{PA} where soundness of the proof is a global property rather than a local property. While this soundness condition has been a key component of previous proof schema constructions, an equivalence between those constructions and an important arithmetic theory has not been shown. The goal of previous work was to provide proof schema with an arithmetic theory rather than providing a significant arithmetic theory with a proof schema formalism. The lack of such a relationship has left the concept of proof schema isolated from the larger community though it has many significant advantages over the \mathbf{LK} -calculus.

What entails from this result is that there is an alternative logical form for proofs of \mathbf{PA} which is a conservative extension of the theory. We can understand the significance of this result by referring to M. Baaz and P. Wojtylak [6] work concerning generalizing proofs in monadic languages. In that work, they discuss at length the results of R.J. Parikh [21] concerning the Kreisel conjecture[5, 18] and the reflection between \mathbf{PA}^* to \mathbf{PA} . The language of \mathbf{PA}^* is similar to \mathbf{PA} except addition and multiplication are represented by ternary predicates where the third component is the result of adding (multiplying) the first two components. Note that such a formalization removes the necessity for binary function symbols.

Reflection principles, as they are discussed in the above mentioned work can be stated as provability of a formula in a given logical form implies that a given formula holds in the theory. If we are considering *analysis* of proofs as defined in [6, 21] then such a provability predicate shows the derivability of infinite families of formula, i.e. analysis drop the term structure of a formula. Let $Prov_{\Pi}$ be a predicate for the given formula A with a given logical form. What the results of the previous section show us is that $Prov_{P\text{-schema}}(A) \leftrightarrow Prov_{\mathbf{PA}}(A)$. While this is already significant, by weakening the logical form of P -schema, that is dropping conservativity, it is possible that new reflection results can be derived.

Let us consider the following P -schema which are related to the work of [6, 21] in that they bound the total proof size. While such P -schema do not have the same properties required of the logical forms for the results [6, 21], that is that there are finitely many matrices of a proof with a bound size, this restriction shows that some interesting properties are easily defined and might pose significant to future work.

Definition 5.1. Let $\Phi = \langle C_1, \dots, C_\alpha \rangle$ be a \mathbf{P} -schema such that $\mathcal{V}_p(es(\Phi)) = \{\alpha\}$. We refer to Φ as *numeric* if there exists a $\kappa \in \mathbb{N}$ such that

$$\max \left\{ |\Phi\{\alpha \leftarrow \beta\}| \mid \beta \in \mathbb{N} \right\} \leq \kappa$$

Numeric \mathbf{P} -schema are not limited to a single passive parameter, the restriction is that every other passive parameter occurring in a numeric \mathbf{P} -schema must be quantified. Going back to section 2.3, this restriction allows us to compute the size modulo the theory T established by the computational sub- \mathbf{P} -schema contained in a given numeric \mathbf{P} -schema.

We leave further investigation of this concept to the wider community and future work.

6 Conclusion

In this paper we generalized the proof schemata formalism of [19] to a much larger fragment of arithmetic. Our extension does not effect the applicability of the proof analysis tools outlined in [14]. Also, we show that our new formalism is at least as expressive as Peano arithmetic. We conjecture that it is a conservative extension of arithmetic and plan to address this question in future work.

Furthermore, we plan to investigate extensions of the method of [19] to this more general formalism being that it provides a cut-elimination complete method for proof schemata. Also, as was addressed in [11], we would like to develop a calculus for construction of \mathbf{P} -schema directly and in the process develop compression techniques for components used in various locations in the same \mathbf{P} -schema. One topic concerning proof schemata which has not been investigated is using inductive definitions other than the natural numbers to index the proof. We plan to investigate generalizations of the indexing sort in future work.

Concerning reflection principles and \mathbf{P} -schema This work illustrates how the formal concept can provide an alternative and advantageous perspective. While we have not used \mathbf{P} -schema to solve a significant problem as of yet, the mere fact that we have shown equivalence to \mathbf{PA} allows further investigation into the use of recursive formalizations. So far most research into schematic formalisms has been focused on proof transformation, the area which gave birth the concept. By providing an equivalence result with a strong arithmetic theory, especially that this equivalence resulted in a conservative reflection principle, we hope that interest in the formalism will grow.

Acknowledgements: We would like to thank Paolo Baldi for introducing us to the Kreisel conjecture. Also, Michael Lettmann, Alexander Leitsch and Mattias Baaz for contributing to the polishing of the work.

References

- [1] Vincent Aravantinos, Ricardo Caferra, and Nicolas Peltier. 2011. Decidability and undecidability results for propositional schemata. *Journal of Artificial Intelligence Research* 40, 1 (Jan. 2011), 599–656. <http://dl.acm.org/citation.cfm?id=2016945.2016961>
- [2] Vincent Aravantinos, Mnacho Echenim, and Nicolas Peltier. 2013. A Resolution Calculus for First-Order Schemata. *Fundamenta Informaticae* (2013).
- [3] Matthias Baaz, Stefan Hetzl, Alexander Leitsch, Clemens Richter, and Hendrik Spohr. 2008. CERES: An analysis of Fürstenberg’s proof of the infinity of primes. *Theoretical Computer Science* 403, 2-3 (Aug. 2008), 160–175. <https://doi.org/10.1016/j.tcs.2008.02.043>
- [4] Matthias Baaz and Alexander Leitsch. 2000. Cut-elimination and redundancy-elimination by resolution. *Journal of Symbolic Computation* 29 (2000), 149–176.
- [5] M. Baaz and P. Pudlák. 1993. Kreisel’s Conjecture for $L\exists_1$. In *Arithmetic, Proof Theory and Computational Complexity*, P. Clote and J. Krajíček (Eds.). Oxford University Press, 29–59. With a postscript by G.ÉJKreisel.
- [6] Matthias Baaz and Piotr Wojtylak. 2008. Generalizing proofs in monadic languages. *Annals of Pure and Applied Logic* 154, 2 (2008), 71–138.
- [7] James Brotherston. 2005. Cyclic Proofs for First-Order Logic with Inductive Definitions. In *Tableaux’05. Lecture Notes in Comp. Sci.*, Vol. 3702. 78–92. https://doi.org/10.1007/11554554_8
- [8] James Brotherston and Alex Simpson. 2010. Sequent calculi for induction and infinite descent. *Journal of Logic and Computation* 21, 6 (2010), 1177–1216.
- [9] David M. Cerna. 2015. *Advances in schematic cut elimination*. Ph.D. Dissertation. Technical University of Vienna. <http://media.obvsg.at/p-AC12246421-2001>.
- [10] David M. Cerna and Alexander Leitsch. 2016. Schematic Cut Elimination and the Ordered Pigeonhole Principle. In *Automated Reasoning - 8th International Joint Conference, IJCAR, 2016, Coimbra, Portugal, June 27 - July 2, 2016, Proceedings*. 241–256. https://doi.org/10.1007/978-3-319-40229-1_17
- [11] David M. Cerna and Michael Lettmann. 2017. Integrating a Global Induction Mechanism into a Sequent Calculus. In *TABLEAUX’17 (Lecture Notes in Comp. Sci.)*.
- [12] David M. Cerna and Michael Lettmann. 2017. Towards a Clausal Analysis of Proof Schemata. In *SYNASC’17 (IEEE Xplorer)*. IEEE.
- [13] Haskell Curry. 1941. A Formalization of Recursive Arithmetic. *American Journal of Mathematics* (1941).
- [14] Cvetan Dunchev, Alexander Leitsch, Mikheil Rukhaia, and Daniel Weller. 2013. Cut-Elimination and Proof Schemata. In *Logic, Language, and Computation*. 117–136.
- [15] Gerhard Gentzen. 1969. Fusion of Several Complete Inductions. In *The Collected Papers of Gerhard Gentzen*, M.E. Szabo (Ed.). Studies in Logic and the Foundations of Mathematics, Vol. 55. Elsevier, 309–311. [https://doi.org/10.1016/S0049-237X\(08\)70829-2](https://doi.org/10.1016/S0049-237X(08)70829-2)
- [16] Stefan Hetzl and Tin Lok Wong. 2017. Restricted notions of provability by induction. [abs/1704.01930](https://arxiv.org/abs/1704.01930) (2017). <http://arxiv.org/abs/1704.01930>
- [17] Jeroen Ketema, Jan Willem Klop, and Vincent van Oostrom. 2005. Vicious Circles in Orthogonal Term Rewriting Systems. *Electr. Notes Theor. Comput. Sci.* 124, 2 (2005), 65–77. <https://doi.org/10.1016/j.entcs.2004.11.020>
- [18] G. Kreisel. 1952. On the interpretation of non-finitist proofs-Part II. *Journal of Symbolic Logic* 17, 1 (1952), 43–58.
- [19] Alexander Leitsch, Nicolas Peltier, and Daniel Weller. 2017. CERES for First-Order Schemata. *Journal of Logic and Computation* (2017). <https://doi.org/10.1093/logcom/exx003>.
- [20] Raymond McDowell and Dale Miller. 1997. Cut-elimination for a logic with definitions and induction. *Theoretical Computer Science* 232 (1997).
- [21] R. J. Parikh. 1973. Some Results on the Length of Proofs. *Trans. Amer. Math. Soc.* 177 (1973), pp. 29–36. <http://www.jstor.org/stable/1996581>
- [22] Harold Simmons. 1988. The realm of primitive recursion. *Archive for Mathematical Logic* 27, 2 (01 Sep 1988), 177–188. <https://doi.org/10.1007/BF01620765>
- [23] Gaisi Takeuti. 1975. *Proof Theory*. Studies in logic and the foundations of mathematics, Vol. 81. American Elsevier Pub.