A General Recursive Construction for Schematic Resolution Derivations

David M. Cerna[☆]

Research Institute for Symbolic Computation (RISC), Johannes Kepler University, Linz

Abstract

Proof schemata provide an alternative formalism for handling inductive argumentation, while non-trivially extending *Herbrand's theorem* to a fragment of arithmetic and thus allowing the construction of *Herbrand sequents* and *expansion trees*. Existing proof analysis methods for proof schemata extract an unsatisfiable characteristic formula representing the cut structure of the proof and from its refutation construct a Herbrand sequent. Unfortunately, constructing the refutation is a task which is highly non-trivial. An automated method for constructing such refutations exists, but it only works for a very weak fragment of arithmetic and is hard to use interactively. More expressive yet interactive methods for the formalization of recursive resolution refutation are complex, hard to work with, and still limited to an undesirably weak class of recursion. In this work we note a particular problem with previous methods, namely they mix the recursive structure with the calculus of refutation. Also we present a modular recursive structure independent of the resolution formalism and proof construction. We illustrate the expressive power of the so called *finite saturated tree* formalism by formalizing the Non-injectivity Assertion's schematic refutation (a variant of the infinitary Pigeonhole principle). None of the previously developed formalism are able to formalize this refutation.

Keywords: Resolution, Proof schema, Induction, Recursion, Pigeonhole Principle

[☆]This research is supported by the FWF project P28789-N32.

1. Introduction

Schematic CERES, introduced in [12], provides a method for the extraction of a Herbrand systems (a generalization of Herbrand sequents) from a refutation of a clausal representation of the cut structure of a recursively defined **LK**-proof. The replacement of induction by a recursive proof construction was used for the analysis of the Fürstenberg's proof of the infinitude of primes by Baaz et al. [2]. Since this early work, the concept of recursively defined **LK**-proofs has been formally developed into the concept of a *proof schema* [8, 15]. A formalism based on recursively defined proofs was chosen for the analysis of Fürstenberg's proof given that recursive proof specifications are more suited for proof analysis and transformation than other methods of formalizing induction. For example, higher-order formalizations of the proof resulted in clausal representations of the cut structure too complex for humans and theorem provers alike to analyze.

Essentially, schematic CERES is a generalization of the CERES Method introduced by M. Baaz and A. Leitsch [3] extended to proof schema. The core of the CERES Method is the extraction of a clausal representation of the cut structure from an **LK**-proof containing cut. This clausal representation is always unsatisfiable and can be refuted using an automated theorem prover of your choice. Unlike other methods of cut-elimination, CERES can benefit from refinements developed by the automated theorem proving community. Given a standard **LK**-proof a clausal representation of its cut structure is easily extractable, however, for recursively defined **LK**-proofs an intermediary step is necessary being that one must finitely represent the cut structure for an infinite number of proofs. Though, any instance within this infinite sequence of proofs can be handled as an **LK**-proof using the CERES method. In contrast to early work [12] which attempted to define a recursive clausal representation of the cut structure as a inductive definition of a negation normal form formula [15] has proven superior and has lead to important resolution variants for recursive clause sets [7].

The problem with earlier attempts to formulate the cut structure as a recursive clause set is the complex syntax required for precisely capturing the cut structure and the increasingly more complex refutational calculi [12] for representation of the resolution derivation [6]. These constructions could not be easily expanded to applications beyond cut-elimination. However, inductive definitions of the cut structure can be analyzed within interactive theorem provers and formal systems such as cyclic proofs [4], COQ [11], etc, therein providing foundational and difficult problems for the wider community[1]. For example, the inductive definition we analyze in this work [5] is challenging for both inductive and non-inductive theorem provers (instances in this case). Unfortunately, while such systems can help one understand what is necessary for constructing a refutation of the inductive definition as a clause set, they do not provide a proof normal form allowing construction of a Herbrand system, the whole point of this exercise. Thus, a formal language for representing the refutation of the cut structure as an inductive definition is still necessary and has been recently addressed in [7] by considering structural clausal forms of the inductively defined cut structure. Part of this work is to provide a modular framework to allow extension of the calculus of [7] to more expressive types of recursion.

Earlier attempts to viably represent the recursive refutation of the cut structure accomplished the goal [12] but in a somewhat unsatisfactory way. In [6], this earlier formalization was used, aided by the theorem provers SPASS [21] and Vampire [14], to perform proof analysis of the so call eventually constant schema, i.e. every total decreasing function with a finite range is eventually constant. This proof is significantly simpler than the infinitary pigeonhole principle, i.e. every total function with a finite range maps two numbers to the same value, discussed in [5] as the Non-Injectivity Assertion, yet the proof analysis was already at the boundary of what can be done without the aid of software.

A second approach was taken in [15] where a superposition prover designed for recursively defined formula [1] was used to refute a clausal form of the cut structure. From this refutation a sequence of substitutions is extractable which can be used to construct a Herbrand system. While this system is complete with respect to the expressive power of the superposition prover, i.e. whenever the prover finds a refutation we can extract a Herbrand system, the Σ_1-incompleteness of the prover is a corollary of the clause set presented in [7]. This incompleteness is entailed by the limitations of the prover to loop_1 programs, the example in [7] requires a loop_2 program. The schematic resolution calculus of [12] also fails at formalizing this example being that it is similarly restricted.

Concerning the most recent work [7] an alternative representation of the

[1]Some of the inductive definitions we have extracted from proofs can be found in the TPTP library [18]

cut structure as a structural clause set is presented. The structural clause form allows formula resolution through the introduction of definitional atoms for subformula. Furthermore, a refutation can contain clauses which are not found in the structural clause set. As long as the clauses are derivable from the structural clause set they can be added to a derivation. These properties allow expressive power beyond existing methods. Nonetheless, there are still issues with this approach, all methods discussed so far mix the calculus used to refute a particular representation of the cut structure with the structure of the refutation itself. This is not a property of non-recursive resolution. This is made quite obvious by the example provided in [7] which is quite simple but not refutable by the superposition prover being that it does not fit the rigid structure required.

This issue is not something which can be avoided given that rigid construction is an intrinsic property of recursively defined objects. However, we can avoid defining this structure in such a way that it is dependent on the choice of operations and inferences. To some extent this was done in [7] where a simple resolution calculus is given which is independent of the precise recursive structure, but the only recursive structure provided is based on the calculus of previous work, which as mentioned above is quite inexpressive. In this paper we consider a generic recursive structure which can be used as a skeleton for recursive derivations. This skeleton can be decorated by resolution derivations or any other tree like construction as long as the non-tautological leaves match the skeleton in terms of a very general concept of matching. This provides a general framework for recursive derivation construction which is expressive and flexible enough to formalize the resolution refutation of the Non-injectivity Assertion Schema [5] which not only requires multiple parameters but also mutual recursion. Formalization of this example has been a goal since its development in 2015 and presents a major achievement. The rest of the paper is as follows:

- In Section 2 & 3, we provide the basics of our recursive formalism, proof schema, and cut structure extraction.
- In Section 4, we provide a mathematical proof of the Non-injectivity Assertion using the precise cut construction of our formal proof. Based on this mathematical proof we provide a proof schema of the formal argument. From this proof schema we extract an inductive definition of the cut structure C. We then show how one can refute this induc-

tive definition by proving the sequent $C \vdash$ in the **LK**-calculus using tautological leaves.

- In Section 5, we provide our general recursive structure for resolution refutations and recursive proofs.
- In Section 6, we provide a formal analysis of the inductive definition extracted from the Non-injectivity Assertion schema using the standard resolution inference and our recursive structure.
- In Section 7, we discuss future work, for example extending Herbrand's theorem as discussed in [15] to our recursive structure decorated by **LK**-calculus proofs. In particular extending the concept of *globalization variable* from [7] to the recursive structure presented here.

2. Preliminaries

In this section we introduce the background necessary for comprehending our results. For more details see one of the following papers containing a more complete description of proof schemata [5, 8, 9, 10, 12, 15].

*2.1. The First-Order **LK**-calculus*

We will refrain from discussing the term language of our logic until the next section on proof schema given that there are additional constraints on term construction not present in standard first-order logic. For this section it is enough to assume that terms are constructed in a standard way and there exists special term constants referred to as *eigenvariables* used for *strong quantifier* inferences. For more details see one of the following books on mathematical logic and proof theory [16, 19]. Formulas are built inductively as usual from a countable set $\mathcal{P}$ of predicate atoms using the logical connectives $\neg, \wedge, \vee, \rightarrow, \forall, \exists$.

To prove first-order formulas and construct formal proofs we will use the **LK**-calculus [13]. Also this calculus will provide a foundation for the **LKS**-calculus which extends the **LK**-calculus by *proof links*. The **LK**-calculus is defined for *sequents* which are pairs of multisets of formulas $\Pi \vdash \Delta$ which are to be interpreted as the formula $\bigvee_{G\in\Pi} \neg G \vee \bigvee_{F\in\Delta} F$. When it is necessary to work with the formula interpretation of a sequent S we write $\mathcal{F}(S)$. A derivation of the **LK**-calculus, referred to as an ***LK**-derivation* is a rooted tree of sequents, rooted at a sequent we refer to as the *end sequent*, such that any two adjacent sequents are part of a sound application of one of the inference rules of Figure 1. The sequent which is closer to the root of the tree is referred to as the *main sequent* of the application and the other sequents are referred to as the *auxiliary sequents*. If the leaves of the tree are of the form $A \vdash A$, where A is atomic, we refer to the tree as an ***LK**-proof*.

2.2. The Schematic Language

In addition to the sort of first order terms, which we refer to as the individual sort ι, we require a second sort of *numeric terms* denoted by ω. The numeric sort is also associated with a countable set of variables $\mathcal{V}_\omega$ which we refer to as *parameter symbols*. However, in this work we will only define proof schema over a single parameter for simplicities sake. For information concerning proof schema with multiple parameters and its relationship to Peano arithmetic please see [10]. Numeric terms without parameter symbols will be referred to as *ground terms*. The set of all ground terms is denoted by $\mathcal{G}_\mathbb{N}$ and contains all terms constructed from the signature $\{0, s(\cdot)\}$. When necessary, we write $\mathcal{V}_i(k)$, for $i \in \{\iota, \omega\}$, to denote the set of free variables (parameters) in a term k. Lower-case Greek characters will denote ground numeric terms unless otherwise stated.

An important extension of the first-order language discussed in the previous section is the addition of *defined function symbols* to the individual sort. We refer to this extension as the *schematic first-order language* which allows an (infinite) sequence o first-order terms to be represented finitely. Defined function symbols are primitive recursive definitions which when normalized are terms of the ι sort. We also allow analogous definitions at the formula level which are referred to as *defined predicate symbols*. In order to give a proper formal definition of schematic first-order formula construction we generalize the concept of formula from the previous section to *formula schema*. Essentially, formulas as previously defined are formula schemata and defined predicate symbols are also formula schemata.

Concerning the definitions of defined symbols and their syntactic construction they will be denoted by $\hat{\cdot}$, i.e. $\hat{P}$. We assume that each defined symbol is associated with a set of convergent rewrite rules which in total constitutes our equational theory $\mathcal{E}$. The **LK**-calculus is extended by an inference rule $\mathcal{E}$ (to be introduced shortly) which allows the application of these rewrite rules to formulas and terms. The rewrite rules of $\mathcal{E}$ have a particular shape as follows, $\widehat{f}(\bar{t}) = E$, where $\bar{t}$ contains no defined symbols, and either $\widehat{f}$ is a function symbol and E is a term or $\widehat{f}$ is a predicate symbol and E is a

$$\frac{\Gamma \vdash \Delta}{D, \Gamma \vdash \Delta}\,\text{w:l} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, D}\,\text{w:r}$$

$$\frac{D, D, \Gamma \vdash \Delta}{D, \Gamma \vdash \Delta}\,\text{c:l} \qquad \frac{\Gamma \vdash \Delta, D, D}{\Gamma \vdash \Delta, D}\,\text{d:r}$$

$$\frac{\Gamma \vdash \Delta, D}{\neg D, \Gamma \vdash \Delta}\,\neg\text{:l} \qquad \frac{D, \Gamma \vdash \Delta}{\Gamma \vdash \Delta, \neg D}\,\neg\text{:r}$$

$$\frac{C, \Gamma \vdash \Delta}{C \wedge D, \Gamma \vdash \Delta}\,\wedge\text{:l} \qquad \frac{D, \Gamma \vdash \Delta}{C \wedge D, \Gamma \vdash \Delta}\,\wedge\text{:l}$$

$$\frac{\Gamma \vdash \Delta, C \qquad \Gamma \vdash \Delta, D}{\Gamma \vdash \Delta, C \wedge D}\,\wedge\text{:r}$$

$$\frac{\Gamma \vdash \Delta, C}{\Gamma \vdash \Delta, C \vee D}\,\vee\text{:r} \qquad \frac{\Gamma \vdash \Delta, D}{\Gamma \vdash \Delta, C \vee D}\,\vee\text{:r}$$

$$\frac{C, \Gamma \vdash \Delta \qquad D, \Gamma \vdash \Delta}{C \vee D, \Gamma \vdash \Delta}\,\vee\text{:l}$$

$$\frac{C, \Gamma \vdash \Delta, D}{\Gamma \vdash \Delta, C \to D}\,\to\text{:r} \qquad \frac{\Gamma \vdash \Delta, C \qquad D, \Gamma \vdash \Delta}{C \to D, \Gamma \vdash \Delta}\,\to\text{:l}$$

$$\frac{\Gamma \vdash \Delta F(\alpha)}{\Gamma \vdash \Delta, \forall x F(x)}\,\forall\text{:r} \qquad \frac{F(t)\Gamma \vdash \Delta}{\forall x F(x), \Gamma \vdash \Delta}\,\forall\text{:l}$$

$$\frac{\Gamma \vdash \Delta F(t)}{\Gamma \vdash \Delta, \exists x F(x)}\,\exists\text{:r} \qquad \frac{F(\alpha)\Gamma \vdash \Delta}{\exists x F(x), \Gamma \vdash \Delta}\,\exists\text{:l}$$

$$\frac{\Gamma \vdash \Delta, C \qquad C, \Gamma \vdash \Delta}{\Gamma \vdash \Delta}\,\text{cut}$$

Figure 1: The **LK**-calculus. Note that α denotes an eigenvariable.

formula schema. The rules can be applied in both directions, i.e. the $\mathcal{E}$-rule is reversible. Such defined function symbols can be defined for both the ω and ι sort, but for the scope of this work we will only consider the defined symbols of the ι sort.

Example 1. *Iterated versions of $\vee$ and $\wedge$ operators (the defined predicates are abbreviated as $\bigvee$ and $\bigwedge$) can be defined using the following equational theory:*

$$\bigvee_{i=0}^{0} P(i) = P(0) \qquad \bigwedge_{i=0}^{0} P(i) = P(0)$$

$$\bigvee_{i=0}^{s(y)} P(i) = \bigvee_{i=0}^{y} P(i) \vee P(s(y)) \qquad \bigwedge_{i=0}^{s(y)} P(i) = \bigwedge_{i=0}^{y} P(i) \wedge P(s(y)).$$

We can also iterate function symbols of the ι sort as follows:

$$It(s(n), a) = f(It(n, a)) \qquad It(0) = a$$

where f is a 1-ary function and a a constant of the ι sort.

We can now generalize the concept of sequent to that of *schematic sequent* which is a pair of multisets of formula schemata Δ, Π denoted by $\Delta \vdash \Pi$. We will denote multisets of formula schemata by upper-case Greek letters. The interpretation remains the same as for sequents, but an additional rule is necessary in order to deal with defined symbols.

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

Note that definitions of **LKE**-derivations and **LKE**-proofs do not differ from those of **LK**. The **LKE**-calculus is the foundation of **LKS**-calculus which we introduce shortly.

Let $S(\overline{x})$ be a schematic sequent and $\overline{x}$ a vector of free variables of the appropriate sort, then $S(\overline{t})$ denotes $S(\overline{x})$ where $\overline{x}$ is replaced by $\overline{t}$, where $\overline{t}$ is a vector of terms of appropriate sort. We denote by the following construction

$$\frac{(\varphi, \bar{t})}{S(\bar{t})}$$

where $S(\overline{x})$ is a schematic sequent and φ a *proof symbol* from the countably infinite set of proof symbols $\mathcal{B}$, a so called *proof link*. Note that we will follow the convention that the left most argument of the proof link is of the ω sort and is the only argument which can contain a parameter symbol. Proof links are to be interpreted as 0-ary inference rules acting as placeholder for proofs. The sequent calculus **LKS** consists of the rules of **LKE** where the leaves of the proof tree can be axioms or proof links. Note that proof links essentially allow arbitrary leaves in the proof tree and do not differ much from **LKE**-derivations. It is the addition of the proof schema construction of the following section which provides a soundness condition for **LKS**-derivations.

3. Proof Schemata

Proof schemata are a sequences of *proof schema components* with a restriction on the occurances of proof links within the proof schema components. Note that proofs may have free parameters, that is parameters are variables of the numeric sort. In this section, we limit the number of parameters to one (as in earlier work [12, 15]). For those interested in the multiple parameter case, see [10].

Definition 2 (Proof schema component). *Let ψ be a proof symbol and n a parameter. A* proof schema component $\mathbf{C}$ *is a triple (ψ, π, ν) where π is a parameter-free **LKE**-proof and ν is an **LKS**-proof containing the parameter n and possibly containing proof links. The end-sequents of the proofs are $S(0, \overline{x})$ and $S(n+1, \overline{x})$, respectively. Given a proof schema component $\mathbf{C} = (\psi, \pi, \nu)$ we define $\mathbf{C}.1 = \psi$, $\mathbf{C}.2 = \pi$, and $\mathbf{C}.3 = \nu$.*

Definition 3. *Let $\mathbf{C}$ and $\mathbf{D}$ be proof schema components such that $\mathbf{C}.1$ is distinct from $\mathbf{D}.1$. We say $\mathbf{C} \succ \mathbf{D}$ if there are no links in $\mathbf{D}.3$ to $\mathbf{C}.1$ and all links in $\mathbf{C}.3$ to $\mathbf{C}.1$ or $\mathbf{D}.1$ are of the following form:*

$$\frac{(\mathbf{C}.1, n, \overline{r})}{S(n, \overline{r})} \qquad\qquad \frac{(\mathbf{D}.1, t, \overline{r})}{S'(t, \overline{r})}$$

for t s.t. $\mathcal{V}(t) \subseteq \{n\}$, and $\overline{r}$ is a vector of terms of the appropriate sort. $S(x)$ and $S'(x)$ are the end sequents of components $\mathbf{C}$ and $\mathbf{D}$, respectively.

Furthermore, let Ψ *be a set of proof schema components containing* $\mathbf{C}$ *and* $\mathbf{D}$*, we say* $\mathbf{C} \succ_{\Psi} \mathbf{D}$ *if* $\mathbf{C} \succ \mathbf{D}$ *and for all proof schema components* $\mathbf{E}$ *of* Ψ *such that* $\mathbf{D} \succ_{\Psi} \mathbf{E}$*,* $\mathbf{C} \succ \mathbf{E}$ *.*

Definition 4 (Proof schema [12]). *Let* $\mathbf{C}_1, \cdots, \mathbf{C}_m$ *be a sequence proof schema components s.t the* $\mathbf{C}_i.1$ *are distinct. Let the end sequents of* $\mathbf{C}_1$ *be* $S(0, \overline{x})$ *and* $S(n+1, \overline{x})$*. We define* $\Psi = \langle \mathbf{C}_1, \cdots, \mathbf{C}_m \rangle$ *as a* proof schema *if* $\mathbf{C}_1 \succ_{\Psi} \ldots \succ_{\Psi} \mathbf{C}_m$*. We call* $S(n+1, \overline{x})$ *the end sequent of* Ψ*.*

Example 2 (Proof schema). *Let us define a proof schema* $\langle (\varphi, \pi, \nu(k)) \rangle$ *with end sequent (schema)*

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

using the equational theory:

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

π *is as follows:*

$$\dfrac{\dfrac{P(0) \vdash P(0) \qquad P(1) \vdash P(1)}{P(0), P(0) \to P(1) \vdash P(1)} \to: l}{P(0), \bigwedge_{i=0}^{0} P(i) \to P(i+1) \vdash P(1)} \mathcal{E}$$

ν *is as follows:*

$$\dfrac{\dfrac{\dfrac{\varphi(n)}{P(0), \bigwedge_{i=0}^{n}(P(i) \to P(i+1)) \vdash P(n+1)} \qquad \dfrac{(\nu_2)}{S(\nu_2)}}{P(0), \bigwedge_{i=0}^{n}(P(i) \to P(i+1)), P(n+1) \to P(n+2) \vdash P(n+2)} cut}{\dfrac{\ldots \mathcal{E} \text{ and } c: l \ldots}{P(0), \bigwedge_{i=0}^{n+1}(P(i) \to P(i+1)) \vdash P(n+2)}}$$

where

$$S(\nu_2) \equiv P(n+1), P(n+1) \to P(n+2) \vdash P(n+2),$$

and ν_2 *is*

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

Notice that proof schemata are essentially primitive recursive **LK**-proofs. When they are evaluated and nominalized for a given numeric term α the result is an **LK**-proof. The evaluation procedure is covered in more detail in [15], here we cover the concepts necessary for the rest of this paper. Evaluation is defined as follows:

Definition 5. *Let Φ be a proof schema of the form $\langle C^1, \cdots C^n \rangle$. For each component C^i, for $1 \leq i \leq n$, we define the following rewrite rules:*

$$C^i.1(0, x_1, \cdots, x_m) \Rightarrow C^1.2 \qquad C^i.1(s(k), x_1, \cdots, x_m) \Rightarrow C^1.3$$

where the end sequent of C^i is $S(s(k), x_1, \cdots, x_m)$ and k is a parameter symbol. The rewrite system for proof links *is the union of these rewrite rules. Furthermore, for a ground term γ, $C^i.1 \Downarrow_\gamma$ is a normal form of the proof $C^i.1(\gamma, x_1, \cdots, x_m)$ under these rewrite rules together with $\mathcal{E}$, the equational theory associated with the proof schema. Further, we define $\Phi \Downarrow_\gamma$ to be $C^1.1 \Downarrow_\gamma$.*

Notice that the rewrite system for proof links within a given proof schema is essentially a set of primitive recursive definitions. The following lemma is a trivial consequence of this observation.

Lemma 1. *The rewrite system for proof links is strongly normalizing, and $\Phi \Downarrow_\gamma$ is an **LK**-proof for all ground numeric terms γ.*

Concerning Lemma 1 we are also considering the normalization of the equational theory $\mathcal{E}$ which as defined above is also strongly normalizing and results in proof normal forms which are constructable with the **LK**-caculus, i.e. no links or $\mathcal{E}$ rules. Thus, this statement can be strengthen to the following theorem.

Theorem 1. *Let Φ be a proof schema with end-sequent $S(n+1, x_1, \cdots, x_m)$. Then $\Phi \Downarrow_\gamma$ is a proof of the sequent $S(\gamma, x_1, \cdots, x_m)$ normalized over $\mathcal{E}$.*

We will not go into anymore detail concerning the construction and properties of proof schemata being that this is enough to understand the rest of this paper. However we have yet to introduce the characteristic formula schema which is an essential concept for the comprehension of our main result.

3.1. Characteristic Formula Extraction

Given a skolemized proof schema[2] Φ which includes cuts one can extract a *schematic characteristic formula* representing the recursive cut structure of Φ. Extraction of a characteristic formula is at the heart of both the CERES method and schematic CERES method as presented in [15]. The precise construction of the characteristic formula is dependent on the cut-status (that is whether or not a formula occurrence is an ancestor of a cut formula) of a given formula occurrence in a proof. To define a schematic version of the characteristic formula we also need to account for formula occurrences derived from previous recursive calls. In some cases these occurrences might be cut ancestors. To capture the cut-status of formula occurrences through proof links, that is through recursive calls, in addition to checking cut ancestor status we add *configurations* of formula occurrences which are treated similarly to cut ancestors. When necessary we denote the current configuration by Ω.

As was done for proof schema evaluation we need to introduce a so called *characteristic formula symbol* which is dependent on a particular configuration of the end sequent of a given proof schema component. This dependence is denoted using the proof symbol associated with the given component. These will be used to define rewrite rules for evaluation and normalization of characteristic formulas. In more precise terms, for each proof schema component C of a proof schema Φ whose proof symbol is ψ, we define a *characteristic formula symbol* $\hat{\Theta}_{\psi,\Omega}$ where Ω is a configuration of the formula occurrences of the end sequent of C. Note that $\hat{\Theta}_{\psi,\Omega}$ will be treated similarly to a defined predicate symbol so that it can occur within the characteristic formula construction. The intended semantics of $\hat{\Theta}_{\psi,\Omega}(t_1, \cdots, t_m)$ is the characteristic formula of the component whose proof symbol is ψ over the terms $t_1, \cdots, t_m$ with configuration Ω. The terms $t_1, \cdots, t_m$ denote arguments to the free variables for the corresponding proof schema component.

Definition 6. *Let π be an* ***LKS****-proof and Ω a configuration of π. Let C be the set of occurrences of cut-formulas in π and $\Xi = \Omega \cup C$. Let ρ be an inference in π. We define the formula $\Theta\rho(\pi, \Omega)$, inductively as follows:*

- *if ρ is an axiom A then $\Theta\rho(\pi, \Omega) = \mathcal{F}(\Xi(A))$*[3]

[2]By skolemized we are referring to proof schemata such that $\forall : r$ and $\exists : l$ inferences have cut-ancestors as their main formulas.

[3]$\Xi(A)$ denotes the sequent of formulas which occur in both A and Ξ

- *if ρ is a proof link of the form $(\hat{\phi}(t1, \cdots, t_n)$ with conclusion S then define Ω_0 as the set of formula occurrences from $\Xi(S)$ and $\Theta\rho(\pi, \Omega) = \hat{\Theta}_{\phi,\Omega_0}(t_1, \cdots, t_m)$.*

- *if ρ is a unary rule with immediate predecessor ρ_0, then $\Theta\rho(\pi, \Omega) = \Theta\rho_0(\pi, \Omega)$*

- *if ρ is a binary rule with immediate predecessors ρ_1, ρ_2, then*

  - *if the auxiliary formulas of ρ are Ξ-ancestors, then $\Theta\rho(\pi, \Omega) = \Theta\rho_1(\pi, \Omega) \wedge \Theta\rho_2(\pi, \Omega)$*
  - *otherwise $\Theta\rho(\pi, \Omega) = \Theta\rho_1(\pi, \Omega) \vee \Theta\rho_2(\pi, \Omega)$*

Finally, define $\Theta\rho(\pi, \Omega) = \Theta\rho_0(\pi, \Omega)$, where ρ_0 is the last inference of π and $\Theta(\pi) = \Theta(\pi, \emptyset)$. We can lift this definition to a proof schema Φ: let n be a parameter not occurring in Φ, then define $\Theta(\Phi) = \hat{\Theta}_{\psi,\emptyset}(n)$ where ψ is the symbol of the left most proof schema component.

As was done for proof schema we can define an evaluation procedure for characteristic formula consisting of primitive recursive definitions which replace characteristic formula symbols with the corresponding characteristic formula. The resulting formula, when evaluated for a numeral γ is a formula of the first-order language used by the **LK**-calculus.

Essentially, we can extend the equational theory $\mathcal{E}$ of a proof schema $\Phi = \langle C^1, \cdots, C^m \rangle$ by the following rewrite rules:

$$\Theta_{C^i.1,\Omega}(0, x_1, \cdots, x_k) \Rightarrow \Theta(C^i.2, \Omega)$$

$$\Theta_{C^i.1,\Omega}(n+1, x_1, \cdots, x_k) \Rightarrow \Theta(C^i.3, \Omega)$$

Where $1 \leq i \leq m$. We retrain from giving an example here for we are lacking an adequate proof schema from which we can extract a characteristic formula. In the next section we provide a characteristic formula for the Non-injectivity Assertion Schema.

4. The Non-injectivity Assertion Schema

In this section we introduce a variation of the Infinitary Pigeonhole Principle which has been adapted to formalization as a proof schema. The most interesting property of this formalization is that it uses a schema of Π_2-cuts.

The construction is based on the tape proof originally presented by C. Urban in his PhD thesis [20]. We first give an "informal" mathematical statement and proof of the assertion and then provide a proof schema based on this informal proof. We then extract the characteristic formula and provide a proof of unsatisfiability within the **LK**-calculus.

4.1. Informal Statement

The Non-injectivity Assertion can be stated as induction over Theorem 2. In order to prove Theorem 2 we first prove the following lemmata:

Lemma 2. *Given a total function $f : \mathbb{N} \rightarrow \{0, \cdots m-1\}$, where $1 \leq m$, then for all $x \in \mathbb{N}$ there exists a $y \in \mathbb{N}$ such that $x \leq y$ and $f(y) = z$ for some $z \in \{0, \cdots m-1\}$.*

Proof. If we assume the contrary, that is there exists an $x \in \mathbb{N}$ for all $y \in \mathbb{N}$ such that $x \leq y$, then $f(y) \neq z$ for $z \in \{0, \cdots m-1\}$ we reach a contradiction with the definition of f.

Lemma 3. *Let f be defined in Lemma 2, then for $1 \leq k \leq m$ one of the following must hold: for all $x_1 \in \mathbb{N}$ there exists a $y_1 \in \mathbb{N}$ such that $x_1 \leq y_1$ and $f(y_1) = z$ for some $z \in \{0, \cdots k-1\}$, or for all $x_2 \in \mathbb{N}$ there exists a $y_2 \in \mathbb{N}$ such that $x_2 \leq y_2$ and $f(y_2) = z$ for some $z \in \{k, \cdots m-1\}$.*

Proof. Assume the contrary, that is there exists a $x_1 \in \mathbb{N}$ for all $y_1 \in \mathbb{N}$ such that when $x_1 \leq y_1$, then $f(y_1) \neq z$ for $z \in \{0, \cdots k-1\}$, and there exists a $x_2 \in \mathbb{N}$ for all $y_2 \in \mathbb{N}$ such that when $x_2 \leq y_2$ then $f(y_2) \neq z$ for $z \in \{k, \cdots m-1\}$. The former part of the above statement implies that for all y_1, such that $x_1 \leq y_1$, $f(y_1) = z$, where $z \in \{k, \cdots m-1\}$. The latter part of the above statement implies that for all y_2, such that $x_2 \leq y_2$, $f(y_2) = z$, where $z \in \{0, \cdots k-1\}$. However, this implies that the following must hold, for $\gamma = \max(x_1, x_2)$, it is the case that $f(\gamma) = z$ for $z \in \{0, \cdots k-1\}$ and $z \in \{k, \cdots m-1\}$, which violates the fact that f is a function. Thus, we have a contradiction.

Lemma 4. *Given a total function $f : \mathbb{N} \rightarrow \mathbb{N}$ and a particular $z \in \mathbb{N}$, such that for all $x \in \mathbb{N}$ there exists a $y \in \mathbb{N}$ where $x \leq y$ and $f(y) = z$,then there exists $p, q \in \mathbb{N}$ such that $p < q$ and $f(p) = f(q)$.*

PROOF. Let us choose two numbers α_0 and α_1 such that $\alpha_0 < \alpha_1$. If we choose two more numbers n_0 and n_1 such that $\alpha_0 \leq n_0$ and $\alpha_1 \leq n_1$ then it is the case that $f(n_0) = f(n_1) = x$.

Lemma 5. *Let f be defined as in Lemma 2, then at least one of the following must hold: there exists $p, q \in \mathbb{N}$ such that $p < q$ and $f(p) = f(q)$, or for all $x \in \mathbb{N}$ there exists a $y \in \mathbb{N}$ such that $x \leq y$ and $f(y) = z$ for $z \in \{0, \cdots, m-2\}$).*

PROOF. Instantiating the free variable k of Lemma 3 with $m-1$ we get the following statement: either for all $x_1 \in \mathbb{N}$ there exists a $y_1 \in \mathbb{N}$ such that $x_1 \leq y_1$ and $f(y_1) = z$ for $z \in \{0, \cdots m-2\}$, or for all $x_2 \in \mathbb{N}$ there exists a $y_2 \in \mathbb{N}$ such that $x_2 \leq y_2$ and $f(y_2) = m-1$. The latter part of this statement and Lemma 4 entail the lemma.

From now on we will refer to the statement of Lemma 5, namely, either there exists $p, q \in \mathbb{N}$ such that $p < q$ and $f(p) = f(q)$, or for all $x \in \mathbb{N}$ there exists a $y \in \mathbb{N}$ such that $x \leq y$ and $f(y) = z$ for $z \in \{0, \cdots, m-2\}$), as $rr(m)$. And, in general we use $rr(k)$ when $1 \leq k \leq m$.

Lemma 6. *Let f be defined as in Lemma 2, and $1 < k < m$, then if $rr(k)$ holds, then so does $rr(k-1)$.*

PROOF. By Lemma 5 we know that $rr(m)$ holds. Assume that the theorem holds for all $w+1$ such that $k < w+1 \leq m$ and we now show that it holds for w. Taking the right hand side of the statement $rr(w+1)$ and applying Lemma 3 for $k = w$, we repeat the argument of Lemma 5 and derive $rr(w)$.

Theorem 2. *Let f be as defined in Lemma 2, if $rr(m)$ holds then so does $rr(1)$.*

PROOF. Chain implications of Lemma 6 together and apply transitivity.

Corollary 1. *Let f be defined as in Lemma 2, then*

$$\exists p, q \in \mathbb{N}(p < q \land f(p) = f(q))$$

PROOF. Apply Lemma 4 to Theorem 2.

4.2. Formal Proof

The proof provided in Section 4.1 is essentially a skeleton of the formal proof schema we present in this section. To prove the Non-injectivity Assertion we need to add additional theory concerning basic equational reasoning, properties of the maximum function, and properties of the linear ordering of the natural numbers. We add this theory as an axiomatic extension, i.e. additional non-tautological initial sequents. Furthermore, certain predicate and function symbols are interpreted over a particular sort as follows $=: \omega \to \omega \to o$, $\leq: \iota \to \iota \to o$, $<: \iota \to \iota \to o$, $f : \iota \to \omega$, $max : \iota \to \iota \to \iota$, and $s : \iota \to \iota$. Note that n in the following sequence of axioms differs from other free variables in that it can only be substituted by numerals, that is $f(0) = f(x), f(0) = f(y) \vdash f(x) = f(y)$ is not an axiom of the theory but $5 = f(x), 5 = f(y) \vdash f(x) = f(y)$ is an axiom. Also, the ι sort is built using the term signature $\{\mathbf{0}, s(\cdot), max(\cdot, \cdot)\}$. The bold zero is used to differentiate from the numeric zero of the ω sort.

$$max(x, y) \leq z \vdash x \leq z \tag{1}$$
$$max(x, y) \leq z \vdash y \leq z \tag{2}$$
$$\vdash x \leq x \tag{3}$$
$$s(x) \leq y \vdash x < y \tag{4}$$
$$n = f(x), n = f(y) \vdash f(x) = f(y) \tag{5}$$
$$\tag{6}$$

We also need an equantional theory defining iterated disjunction which is defined in Example 1. The cut formula contains an instance of iterated disjunction and is as follows:

$$\forall x \exists y \left(x \leq y \wedge \bigvee_{i=0}^{n+1} i = f(y) \right) \tag{7}$$

which is an essential part of $rr(n)$ from Lemma 5. The end sequent only defines that a total function with a finite domain must have two distinct positions which map to the same value in the range.

$$\forall x \left(\bigvee_{i=0}^{n+1} i = f(x) \right) \vdash \exists x, y \, (x < y \wedge f(x) = f(y)) \tag{8}$$

The proof schema formalizing the Non-injectivity Assertion consists of two proof components $\langle(\psi,\pi,\nu),(\varphi,\pi',\nu')\rangle$ where the individual components are defined as follows:

<u>Proof: π</u>

$$
\cfrac{\cfrac{\cfrac{\begin{array}{c}0=f(\alpha),0=f(\beta)\vdash\\ f(\alpha)=f(\beta)\end{array}\qquad\qquad \begin{array}{c}s(\alpha)\leq\beta\vdash\\ \alpha<\beta\end{array}}{\begin{array}{c}0=f(\alpha),s(\alpha)\leq\beta,0=f(\beta)\vdash\\ \alpha<\beta\wedge f(\alpha)=f(\beta)\end{array}}\;\wedge:r}{\textit{unary rules}}}{\begin{array}{c}\forall x\exists y\,(x\leq y\wedge 0=f(y))\vdash\\ \exists x,y\,(x<y\wedge f(x)=f(y))\\ (A)\end{array}}
$$

$$
\cfrac{\cfrac{\cfrac{\cfrac{\vdash\alpha\leq\alpha\qquad\qquad 0=f(\alpha)\vdash 0=f(\alpha)}{0=f(\alpha)\vdash\alpha\leq\alpha\wedge 0=f(\alpha)}\;\wedge:r}{\textit{unary rules}}}{\begin{array}{c}\forall x\left(\bigvee_{i=0}^{0} i=f(x)\right)\vdash\\ \forall x\exists y\,(x\leq y\wedge 0=f(y))\end{array}}\qquad\qquad (\mathrm{A})}{\forall x\left(\bigvee_{i=0}^{0} i=f(x)\right)\vdash\exists x,y\,(x<y\wedge f(x)=f(y))}\;\textit{cut}
$$

<u>Proof: ν</u>

$$
\cfrac{\cfrac{\cfrac{\vdash\alpha\leq\alpha\qquad\qquad \begin{array}{c}\bigvee_{i=0}^{n+1} i=f(\alpha)\vdash\\ \bigvee_{i=0}^{n+1} i=f(\alpha)\end{array}}{0=f(\alpha)\vdash\alpha\leq\alpha\wedge 0=f(\alpha)}\;\wedge:r}{\textit{unary rules}}}{\begin{array}{c}\forall x\left(\bigvee_{i=0}^{n+1} i=f(x)\right)\vdash\\ \forall x\exists y\left(x\leq y\wedge\bigvee_{i=0}^{n+1} i=f(x)\right)\\ (A)\end{array}}
$$

$$
\cfrac{(\mathrm{A})\qquad\qquad \begin{array}{c}\varphi(n+1)\\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\ \forall x\exists y\left(x\leq y\wedge\bigvee_{i=0}^{n+1} i=f(x)\right)\vdash\\ \exists x,y\,(x<y\wedge f(x)=f(y))\end{array}}{\forall x\left(\bigvee_{i=0}^{n+1} i=f(x)\right)\vdash\exists x,y\,(x<y\wedge f(x)=f(y))}\;\textit{cut}
$$

Note that in proof ν of the ψ proof component the initial sequent $\bigvee_{i=0}^{n+1} i = f(\alpha) \vdash \bigvee_{i=0}^{n+1} i = f(\alpha)$ is technically not an initial sequent. Another proof component should be added to the proof schema unrolling this sequent into proper initial sequents. however, given the trivial nature of this statement we found it best to leave out of the complete construction in order to avoid a larger formal proof.

Proof: π'

$$\dfrac{\dfrac{\begin{array}{c}0 = f(\alpha), 0 = f(\beta) \vdash \\ f(\alpha) = f(\beta)\end{array} \qquad \begin{array}{c}s(\alpha) \leq \beta \vdash \\ \alpha < \beta\end{array}}{\begin{array}{c}0 = f(\alpha), s(\alpha) \leq \beta, 0 = f(\beta) \vdash \\ \alpha < \beta \wedge f(\alpha) = f(\beta)\end{array}} \wedge : r}{\dfrac{\textit{unary rules}}{\begin{array}{c}\forall x \exists y \left(x \leq y \wedge \bigvee_{i=0}^{0} i = f(x)\right) \vdash \\ \exists x, y \left(x < y \wedge f(x) = f(y)\right)\end{array}}}$$

Proof: ν'

$$\dfrac{\begin{array}{c}max(\alpha, \beta) \leq \gamma \vdash \\ \alpha \leq \gamma\end{array} \qquad \begin{array}{c}\bigvee_{i=0}^{n} i = f(\gamma) \vdash \\ \bigvee_{i=0}^{n} i = f(\gamma)\end{array}}{\begin{array}{c}max(\alpha, \beta) \leq \gamma, \bigvee_{i=0}^{n} i = f(\gamma) \vdash \\ \alpha \leq \gamma \wedge \bigvee_{i=0}^{n} i = f(\gamma) \\ (C)\end{array}}$$

$$\dfrac{(C) \qquad \dfrac{\begin{array}{c}max(\alpha, \beta) \leq \gamma \vdash \\ \beta \leq \gamma\end{array} \qquad \begin{array}{c}n + 1 = f(\gamma) \vdash \\ n + 1 = f(\gamma)\end{array}}{\begin{array}{c}max(\alpha, \beta) \leq \gamma, n + 1 = f(\gamma) \vdash \\ \beta \leq \gamma \wedge n + 1 = f(\gamma)\end{array}}}{\begin{array}{c}max(\alpha, \beta) \leq \gamma, \bigvee_{i=0}^{n+1} i = f(\beta) \vdash \\ \alpha \leq \gamma \wedge \bigvee_{i=0}^{n} i = f(\gamma), \\ \beta \leq \gamma \wedge n + 1 = f(\gamma) \\ (B)\end{array}}$$

$$
\dfrac{
\dfrac{\dfrac{\begin{array}{c}(B)\\ max(\alpha,\beta)\leq\gamma, \bigvee_{i=0}^{n+1} i=f(\gamma)\vdash\\ \alpha\leq\gamma\wedge\bigvee_{i=0}^{n} i=f(\gamma),\\ \beta\leq\gamma\wedge n+1=f(\gamma)\end{array}}{\textit{unary rules}}}{\begin{array}{c}\forall x\exists y\left(x\leq y\wedge\bigvee_{i=0}^{n+1} i=f(y)\right)\vdash\\ \forall x\exists y\left(x\leq y\wedge\bigvee_{i=0}^{n} i=f(y)\right),\\ \forall x\exists y(x\leq y\wedge n+1=f(y))\end{array}}
\qquad
\begin{array}{c}\varphi(n)\\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\ \forall x\exists y\left(x\leq y\wedge\bigvee_{i=0}^{n} i=f(x)\right)\vdash\\ \exists x,y\left(x<y\wedge f(x)=f(y)\right)\end{array}
}{\begin{array}{c}\forall x\exists y\left(x\leq y\wedge\bigvee_{i=0}^{n+1} i=f(x)\right)\vdash\\ \exists x,y\left(x<y\wedge f(x)=f(y)\right),\\ \forall x\exists y(x\leq y\wedge n+1=f(y))\\ (A)\end{array}}
$$

$$
\dfrac{\dfrac{\dfrac{\begin{array}{c}n+1=f(\alpha),\\ n+1=f(\beta)\vdash\\ f(\alpha)=f(\beta)\end{array}\qquad\begin{array}{c}s(\alpha)\leq\beta\vdash\\ \alpha<\beta\end{array}}{\begin{array}{c}n+1=f(\alpha), s(\alpha)\leq\beta, n+1=f(\beta)\vdash\\ \alpha<\beta\wedge f(\alpha)=f(\beta)\end{array}}\wedge:r}{\textit{unary rules}}}{\begin{array}{c}\forall x\exists y(x\leq y\wedge n+1=f(y))\vdash\\ \exists x,y\left(x<y\wedge f(x)=f(y)\right)\\ (D)\end{array}}
$$

$$
\dfrac{\begin{array}{c}(A)\\ \forall x\exists y\left(x\leq y\wedge\bigvee_{i=0}^{n+1} i=f(x)\right)\vdash\\ \exists x,y\left(x<y\wedge f(x)=f(y)\right),\\ \forall x\exists y(x\leq y\wedge n+1=f(y))\end{array}\qquad (D)}{\begin{array}{c}\forall x\exists y\left(x\leq y\wedge\bigvee_{i=0}^{n+1} i=f(x)\right)\vdash\\ \exists x,y\left(x<y\wedge f(x)=f(y)\right)\end{array}}
$$

4.3. Characteristic Formula

Using the characteristic formula extraction method defined in Section 2 we can extract a characteristic formula from the formal proof schema defined in the previous section. The inductive definition of the characteristic formula requires three additional rewrite rules and two schema of configurations $\Omega(n)$ referring to the occurrence of the previous cut formula:

$$\Omega(n) = \forall x \exists y (x \leq y \wedge \bigvee_{i=0}^{n} i = f(y) \vdash \qquad \Omega'(n) = \vdash \bigvee_{i=0}^{n} i = f(y).$$

As we mentioned in the previous section $\bigvee_{i=0}^{n+1} i = f(\alpha) \vdash \bigvee_{i=0}^{n+1} i = f(\alpha)$ is not a proper initial sequent and should have a proof schema component decomposing it which we give the proof symbol χ.

$$\begin{aligned}
\Theta_{\psi,\emptyset}(s(n)) \Longrightarrow & \; x \leq x \;\wedge\; \Theta_{\psi,\Omega(s(n))}(s(n)) \;\wedge\; \Theta_{\chi,\Omega'(s(n))}(s(n),y) \\
\Theta_{\psi,\emptyset}(0) \Longrightarrow & x \leq x \;\wedge\; \Theta_{\psi,\Omega(0)}(0) \;\wedge\; \Theta_{\chi,\Omega'(0)}(0,y) \\
\Theta_{\varphi,\Omega(s(n))}(s(n)) \Longrightarrow & max(x,y) \leq z \rightarrow x \leq z \;\wedge \\
& max(x',y') \leq z' \rightarrow y' \leq z' \;\wedge \\
& \neg s(y'') \leq x'' \vee \neg s(n) = f(y'') \vee \neg s(n) = f(x''))\wedge \\
& \Theta_{\varphi,\Omega(n)}(n) \\
\Theta_{\varphi,\Omega(0)}(0) \Longrightarrow & \; \neg s(y) \leq x \vee \neg 0 = f(y) \vee \neg 0 = f(x)) \\
\Theta_{\chi,\Omega'(s(n))}(s(n),x) \Longrightarrow & \; s(n) = f(x) \vee \Theta_{\chi,\Omega'(n)}(n,x) \\
\Theta_{\chi,\Omega'(0)}(0,x) \Longrightarrow & \; 0 = f(x)
\end{aligned}$$

Thus the schematic characteristic formula of the Non-injectivity Assertion is $\Theta(\Phi) = \Theta_{\psi,\emptyset}$.

Notice that schematic characteristic formula is in conjunctive normal form and thus can easily be transformed into a clause set. We provide the following simplified representation to aid readability:

$$
\begin{aligned}
\psi(s(n)) &\Longrightarrow \forall x(\ x \leq x\ )\ \wedge\ \varphi(s(n))\ \wedge\ \forall x(\ \chi(s(n),x)\ )\\
\psi(0) &\Longrightarrow \forall x,y(\neg s(y) \leq x \vee \neg 0 = f(y) \vee \neg 0 = f(x)))\ \wedge\ \forall x(\ x \leq x\ )\wedge\\
&\quad\ \forall x(\ \chi(s(n),x))\\
\varphi(s(n)) &\Longrightarrow \forall x,y,z(\ max(x,y) \leq z \rightarrow x \leq z\ )\wedge\\
&\quad\ \forall x,y,z(\ max(x,y) \leq z \rightarrow y \leq z\ )\wedge\\
&\quad\ \forall x,y(\neg s(y) \leq x \vee \neg s(n) = f(y) \vee \neg s(n) = f(x)))\wedge\ \varphi(n)\\
\varphi(0) &\Longrightarrow \forall x,y(\neg s(y) \leq x \vee \neg 0 = f(y) \vee \neg 0 = f(x)))\\
\chi(s(n),a) &\Longrightarrow s(n) = f(a) \vee \chi(n,a)\\
\chi(0,a) &\Longrightarrow 0 = f(a)
\end{aligned}
$$

In the next section we provide a proof that $\psi(n) \downarrow_{\alpha}\vdash$, where α is a numeral, is provable using the **LK**-calculus and only tautological leaves.

4.4. Proof of $\psi(n)$ Unsatisfiability

In this section we provide the recursive constructions allowing one to prove $\psi(n) \downarrow_{\alpha}\vdash$ within the **LK**-calculus for any numeral α. The informal proof we provide in this section is essentially the proof we formalize as a *decorated finite saturated tree* later in this paper.

The term instantiations required to prove $\psi(n) \downarrow_{\alpha}\vdash$ within this calculus are recursively definable and can be presented as the following defined function symbol which we refer to as the *iterated max function*:

$$
\begin{aligned}
m(n+1) =& max(s(m(n)), m(n))\\
m(0) =& max(s(0), 0)
\end{aligned}
$$

In order to provide an argument concerning the provability of $\psi(n) \downarrow_{\alpha}\vdash$ for any α we need a few auxiliary lemmata concerning the construction of iterated max functions within the proof. Note that we will abbreviate the formulas derivable from $\psi(n) \downarrow_{\alpha}\vdash$ by a context variable Δ.

Lemma 7. *Let n be a numeral and $0 \leq k \leq n$ then $\Delta \vdash m(k) \leq m(n)$ and $\Delta \vdash s(0) \leq m(n)$ are provable in **LK** without cut and from tautological initial sequents.*

Proof. If we assume that the lemma holds for $\Delta \vdash max(s(0), 0) \leq m(n)$ it is quite easy to show that the lemma holds for $\Delta \vdash s(0) \leq m(n)$ using the following derivation

$$\dfrac{\dfrac{\begin{array}{c}\vdots \\ \Delta, \vdash max(s(0),0) \le m(n)\end{array} \qquad s(0) \le m(n) \vdash s(0) \le m(n)}{\Delta, max(s(0),0) \le m(n) \to s(0) \le m(n) \vdash s(0) \le m(n)}}{\Delta, \vdash s(0) \le m(n)}$$

This implies that proving the lemma for all other cases proves the lemma for this case as well. Now let us consider the case when $k = n$, that is the sequent:

$$\Delta \vdash m(n) \le m(n)$$

This is trivially true given that Δ contains $\forall x(x \le x)$. Let us now assume that the lemma hold for all k s.t. $0 < k \le n$ and show that the theorem also holds for $0 \le l < k \le n$. Let us assume that $l = k - 1$ w.l.o.g. That is, we assume the following sequent is provable:

$$\Delta \vdash m(k-1) \le m(n)$$

Note that Δ contains the formula $\forall x, y, z(\ max(x, y) \le z \implies y \le z\ )$ which can be instantiated as follows:

$$max(s(m(k-1)), m(k-1)) \le m(n) \to m(k-1) \le m(n))$$

Thus, the sequent $\Delta \vdash m(k-1) \le m(n)$ can be derived from the sequent

$$\begin{array}{c} max(s(m(k-1)), m(k-1)) \le m(n) \to m(k-1) \le m(n)), \\ \Delta \vdash m(k-1) \le m(n) \end{array}$$

Using the following derivation:

$$\dfrac{\dfrac{\text{Induction Hypothesis}}{\Delta \vdash m(k) \le m(n)}}{\Delta \vdash max(s(m(k-1)), m(k-1)) \le m(n)}$$
$$(A)$$

$$\dfrac{(A) \qquad m(k-1) \le m(n)) \vdash m(k-1) \le m(n)}{\begin{array}{c} max(s(m(k-1)), m(k-1)) \le m(n) \to m(k-1) \le m(n)), \Delta \vdash \\ m(k-1) \le m(n) \end{array}}$$

Lemma 7 provides a method for deriving sequents of the form $\Delta \vdash m(k) \leq m(n)$. Note that we only used one of the two formulas defining *max* in the derivation of Lemma 7. The other *max* defining formula is used in the following lemma for the derivation of sequents $\Delta \vdash s(m(k)) \leq m(n)$ which are essential to our main unsatisfiability argument:

Lemma 8. *Let n be a numeral and $0 \leq k < n$ then $\Delta \vdash s(m(k)) \leq m(n)$ is provable in **LK** without cut and from tautological initial sequents.*

Proof. Rather than a proof by induction, we can prove the statement by a case distinction on k. First we consider the case $k = n-1$, that is the sequent:

$$\Delta \vdash s(m(n-1)) \leq m(n)$$

Using the formula $\forall x, y, z(\ max(x,y) \leq z \rightarrow x \leq z \ )$ instantiated as follows:

$$max(s(m(n-1)), m(n-1)) \leq m(n) \rightarrow s(m(n-1)) \leq m(n))$$

we can construct the following derivation:

$$\dfrac{\dfrac{\vdash m(n) \leq m(n)}{\vdash max(s(m(n-1)), m(n-1)) \leq m(n)} \qquad \begin{array}{c} s(m(n-1)) \leq m(n)) \vdash \\ s(m(n-1)) \leq m(n) \end{array}}{\begin{array}{c} max(s(m(n-1)), m(n-1)) \leq m(n) \rightarrow s(m(n-1)) \leq m(n)), \Delta \vdash \\ s(m(n-1)) \leq m(n) \end{array}}$$

When $0 \leq k < n-1$ we use Lemma 7 as an assumption in the derivation. Once again we instantiate $\forall x, y, z(\ max(x,y) \leq z \rightarrow x \leq z \ )$ as follows:

$$max(s(m(k)), m(k)) \leq m(n) \rightarrow s(m(k)) \leq m(n))$$

resulting in the derivation:

$$\dfrac{\dfrac{\dfrac{\text{Lemma 7}}{\Delta \vdash m(k+1) \leq m(n)}}{\Delta \vdash max(s(m(k)), m(k)) \leq m(n)} \qquad \begin{array}{c} s(m(k)) \leq m(n)) \vdash \\ s(m(k)) \leq m(n) \end{array}}{\begin{array}{c} max(s(m(k)), m(k)) \leq m(n) \rightarrow s(m(k)) \leq m(n)), \Delta \vdash \\ s(m(k-1)) \leq m(n) \end{array}}$$

Lemmata 7 & 8 provide a method for constructing the necessary term instantiations within the derivation. The final step in the argument concerning provability of $\psi(n) \downarrow_\alpha \vdash$ is to instantiate the schematic length clause constructed by χ, by the correct terms. We will refrain form replacing n by the numeral α from now on as the proof is clearer when $n+1$ is used instead. At the moment the following choice of instantiations might seem a bit contrived, but its purpose will become clear once we present the derivation:

$$\Delta, \left(\bigwedge_{i=0}^{n+1} \bigvee_{j=0}^{n+1} j = f(m(i)) \right), \bigvee_{j=0}^{n+1} j = f(0) \vdash \tag{9}$$

If we isolate the disjunction $\bigvee_{j=0}^{n+1} j = f(m(n+1))$ of Equation 9 and consider applications of the $\vee : l$ inference rule where the disjunction is the main formula, the following sequent is clearly an ancestor of Equation 9 :

$$\Delta, \left(\bigwedge_{i=0}^{n} \bigvee_{j=0}^{n+1} j = f(m(i)) \right), \bigvee_{j=0}^{n+1} j = f(0), k = f(m(n+1)) \vdash \tag{10}$$

for $0 \leq k \leq n+1$. Note that the formula

$$\forall x, y(\neg s(x) \leq y \vee \neg k = f(x) \vee \neg k = f(y)) \tag{11}$$

is stored within the context Δ. Thus, from Equation 10 the following two sequents are derivable:

$$\begin{gathered} \Delta, \neg s(m(n)) \leq m(n+1) \vee \neg k = f(m(n)) \vee \neg k = f(m(n+1)), \\ k = f(m(n+1)), k = f(m(n)) \vdash \end{gathered} \tag{12}$$

$$\begin{gathered} \Delta, \left(\bigwedge_{i=0}^{n-1} \bigvee_{j=0}^{n+1} j = f(m(i)) \right), \\ \left(\bigvee_{j=0}^{k-1} j = f(m(n)) \vee \bigvee_{j=k+1}^{n+1} j = f(m(n)) \right), \\ \left(\bigvee_{j=0}^{n+1} j = f(0) \right), k = f(m(n+1)) \vdash \end{gathered} \tag{13}$$

Obviously, Equation 12 is provable with the help of Lemmata 7 & 8. Equation 13 is part of a recursive pattern essential to our proof and ending at the following ancestor sequent:

$$\begin{gathered} \Delta, \bigwedge_{i=0}^{n} \left(\bigvee_{j=0}^{k-1} j = f(m(i)) \vee \bigvee_{j=k+1}^{n+1} j = f(m(i)) \right), \\ \left(\bigvee_{j=0}^{k-1} j = f(0) \vee \bigvee_{j=k+1}^{n+1} j = f(0) \right) \vdash \end{gathered} \tag{14}$$

Note that, if $k = n + 1$ then this sequent is equivalent to

$$\Delta, \bigwedge_{i=0}^{n} \left(\bigvee_{j=0}^{n} j = f(m(i)) \right), \left(\bigvee_{j=0}^{n} j = f(0) \right) \vdash \tag{15}$$

Notice that Equation 15 is precisely the induction hypothesis if we where to perform induction over the value of n on Equation 9. Of course, we cannot perform this derivation for $k = n + 1$ only and thus must consider appropriate renamings of the ω sort in order to take advantage of this symmetry. Fortunately, the formal proof presented later in this paper does not need to consider these renamings.

For the following theorem we take advantage of the symmetry pointed out in Equation 15 and assume that an appropriate renaming of the ω-sort terms can be performed in order to maintain a sound derivation.

Theorem 3. *For $0 \leq n$, the sequent*

$$\Delta, \left(\bigwedge_{i=0}^{n} \bigvee_{j=0}^{n} j = f(m(i)) \right), \bigvee_{j=0}^{n} j = f(0) \vdash$$

*is provable in **LK** without cut and from tautological initial sequents.*

Proof. Let us assume as the basecase that $n = 0$. Then we are deriving the sequent $\Delta, 0 = f(m(0)), 0 = f(0) \vdash$. By an appropriate renaming of the terms of the ω sort the sequent we are deriving is $\Delta, k = f(m(0)), k = f(0) \vdash$ for $0 \leq k \leq m$ where m is a predetermined maximum value. By construction, Δ includes the formula

$$\forall x, y(\neg s(x) \leq y \vee \neg k = f(x) \vee \neg k = f(y))$$

and thus the case when $n = 0$ is derivable in the **LK**-calculus. Let us assume that the theorem holds for all $0 \leq m \leq n$ and show that it holds for $n + 1$. Let us start by considering the sequent of the induction hypothesis:

$$\Delta, \left(\bigwedge_{i=0}^{n} \bigvee_{j=0}^{n} j = f(m(i)) \right), \bigvee_{j=0}^{n} j = f(0) \vdash$$

By an appropriate renaming of the terms of the ω sort and an arbitrary $0 \leq k \leq n + 1$ we can transform this sequent into the following:

$$\begin{aligned}\Delta, \bigwedge_{i=0}^{n} \left(\bigvee_{j=0}^{k-1} j = f(m(i)) \vee \bigvee_{j=k+1}^{n+1} j = f(m(i)) \right), \\ \left(\bigvee_{j=0}^{k-1} j = f(0) \vee \bigvee_{j=k+1}^{n+1} j = f(0) \right) \vdash\end{aligned} \tag{16}$$

Furthermore for $0 \leq r < n+1$, every sequent of the form

$$\Delta, k = f(m(n+1)), k = f(m(r)) \vdash \tag{17}$$

is derivable. Using Equation 16 and Equation 17 for all $0 \leq r < n+1$ as the auxiliary sequents of $\vee : l$ inference rules we can derive the following sequent:

$$\begin{aligned}\Delta, \bigwedge_{i=0}^{n} \left(\bigvee_{j=0}^{n+1} j = f(m(i)) \right), \\ \left(\bigvee_{j=0}^{k-1} j = f(0) \vee \bigvee_{j=k+1}^{n+1} j = f(0) \right), k = f(m(n+1)) \vdash\end{aligned} \tag{18}$$

From Equation 18 and the sequent

$$\Delta, k = f(m(n+1)), k = f(0) \vdash \tag{19}$$

we need to make one last inference step resulting in

$$\Delta, \bigwedge_{i=0}^{n} \left(\bigvee_{j=0}^{n+1} j = f(m(i)) \right), \left(\bigvee_{j=0}^{n+1} j = f(0) \right), k = f(m(n+1)) \vdash \tag{20}$$

At this point we need to start a second induction whose basecase is

$$\Delta, \bigwedge_{i=0}^{n} \left(\bigvee_{j=0}^{n+1} j = f(m(i)) \right), \left(\bigvee_{j=0}^{n+1} j = f(0) \right), 0 = f(m(n+1)) \vdash \tag{21}$$

We assume as the induction hypothesis that the following sequent is provable

$$\begin{aligned}\Delta, \bigwedge_{i=0}^{n} \left(\bigvee_{j=0}^{n+1} j = f(m(i)) \right), \\ \left(\bigvee_{j=0}^{n+1} j = f(0) \right), \bigvee_{j=0}^{n} j = f(m(n+1)) \vdash\end{aligned} \tag{22}$$

and show that

$$\begin{aligned}\Delta, \bigwedge_{i=0}^{n} \left(\bigvee_{j=0}^{n+1} j = f(m(i)) \right), \\ \left(\bigvee_{j=0}^{n+1} j = f(0) \right), \bigvee_{j=0}^{n+1} j = f(m(n+1)) \vdash\end{aligned} . \tag{23}$$

This trivially follows from the deriability of

$$\Delta, \bigwedge_{i=0}^{n} \left(\bigvee_{j=0}^{n+1} j = f(m(i)) \right), \left(\bigvee_{j=0}^{n+1} j = f(0) \right), n+1 = f(m(n+1)) \vdash \ .$$

Note that Equation 23 is equivalent to

$$\Delta, \bigwedge_{i=0}^{n+1} \left(\bigvee_{j=0}^{n+1} j = f(m(i)) \right), \left(\bigvee_{j=0}^{n+1} j = f(0) \right) \vdash \qquad (24)$$

the precise theorem we where attempting to prove.

This proof gives an outline of the proof we formalize in the coming sections without the heavy syntax required by our general recursive framework.

5. A General Recursive Structure for Schema

In this section we introduce our general recursive structure for the formalization of proof and resolution derivation schema. Our goal is to keep the formal construction as modular as possible, that is independent of the particular calculus. We first define the concept of finite saturated trees and then show how one can decorate a particular finite saturated tree with resolution derivations in such a way that the resolution refutation of the Non-Injectivity Assertion schema is constructed. Towards this goal we have to generalize the ω sort beyond numerals.

5.1. Point Spaces and Curves

The basic term construction used for finite saturated trees are primitive recursive functions defined on numerals. We use the following definition for this work.

Definition 7. *We define the set $\mathcal{PR}$ of* primitive recursive functions, *$f : \mathbb{N} \rightarrow \mathbb{N}$, as follows:*

(1) $0 \in \mathcal{PR}$

(2) if $x \in \mathcal{PR}$, then $s(x) \in \mathcal{PR}$

(3) If $x_1, \cdots, x_n \in \mathcal{PR}$ then we define $P_i^n(x_1, \cdots, x_n) = x_i$, for each $1 \leq i \leq n$. $P_1^n, \cdots, P_n^n \in \mathcal{PR}$

(4) If $x_1, \cdots, x_n, g_1, \cdots, g_k, f \in \mathcal{PR}$ such that $g_1, \cdots, g_k$ are n-ary and f is k-ary then the function $f(g_1(x_1, \cdots, x_n), \cdots, g_k(x_1, \cdots, x_n)) \in \mathcal{PR}$

(5) If $x_1, \cdots, x_n, g, f \in \mathcal{PR}$ such that g is n-ary and f is $n+2$-ary then the function

$$h(0, x_1, \cdots, x_n) = g(x_1, \cdots, x_n)$$
$$h(s(k), x_1, \cdots, x_n) = f(k, h(k, x_1, \cdots, x_n), x_1, \cdots, x_n)$$

is in $\mathcal{PR}$

The functions constructed using Definition 7 are the objects constituting *term spaces* which we define shortly. We assume a countable set of parameter symbols $\mathcal{P} = \{k_i\}_{i=0}^{\infty}$ used for term construction. Terms constructed from cases (1) and (2) of Definition 7 which are parameter free will be referred to as *ground terms* $\mathcal{G}$, that is all terms built from the signature $\{s(\cdot), 0\}$. We will denote ground terms by overline lowercase latin characters, i.e. $\overline{n}$. The set of terms constructible from the signature $\mathcal{PR}$ together with $\mathcal{P}$ will be referred to as $\mathcal{T}$. In particular we will consider spaces of m-tuples of terms denoted by $\mathcal{T}^m$ for $m \in \mathbb{N}$ which we will refer to as *term spaces*. Term spaces are assumed to be ordered by a well-founded ordering $\preceq_m$ (by $\prec_m$ we are referring to the irreflexive version of $\preceq_m$). Note that by $\preceq_m$ we mean there is an ordering for each m where m is the dimension of a term space. When representing a particular tuple $v \in \mathcal{T}^m$ we will write $v = |n_1, \cdots, n_m|$. Let $v \in \mathcal{T}^m$ and $t \in \mathcal{T}$, by $\mathcal{V}(v)$ and $\mathcal{V}(t)$ we refer to the parameters of $\mathcal{P}$ used in the construction of v and t.

We need to consider paths through a term space $\mathcal{T}^m$. This implies some sort of external ordering/mechanism for deciding the next point in a term space which is on a path. In order to construct such paths, we define *path symbols* $\mathcal{R} = \{\rho_i\}_{i=0}^{\infty}$ which are ordered by an independent well-founded ordering $\preceq_R$ (by $\prec_R$ we are referring to the irreflexive version of $\preceq_R$) , that is independent from the well founded ordering on the associated term space.

Each path symbol in $\mathcal{R}$ is associated with a term space of a fixed size and a list of so called "scratch pad" terms of a fixed size (This idea is inspired by [17]). We define a total function, referred to as an *arity function* $\mathcal{A} : \mathcal{R} \to (\mathbb{N}, \mathbb{N})$, mapping each path symbol to a pair of natural numbers such that $\mathcal{A}(x, 0)$ and $\mathcal{A}(x, 1)$ refer to the first and second value in the pair $\mathcal{A}(x)$, respectively. The value of $\mathcal{A}(x, 0)$ is the size of the term space associated with x and $\mathcal{A}(x, 1)$ is the size of the scratch pad.

Definition 8. *The* point space $\mathcal{P}^\star(\mathcal{A})$ over an arity function $\mathcal{A}$ *is the space of all 3-tuples of the form $(\rho, v, \{a_1, \cdots a_{\mathcal{A}(\rho,1)}\})$, where $\rho \in \mathcal{R}$, $v \in \mathcal{T}^{\mathcal{A}(\rho,0)}$ and $a_1, \cdots, a_{\mathcal{A}(\rho,1)} \in \mathcal{T}$. We refer to the 3-tuples of $\mathcal{P}^\star(\mathcal{A})$ as* points.

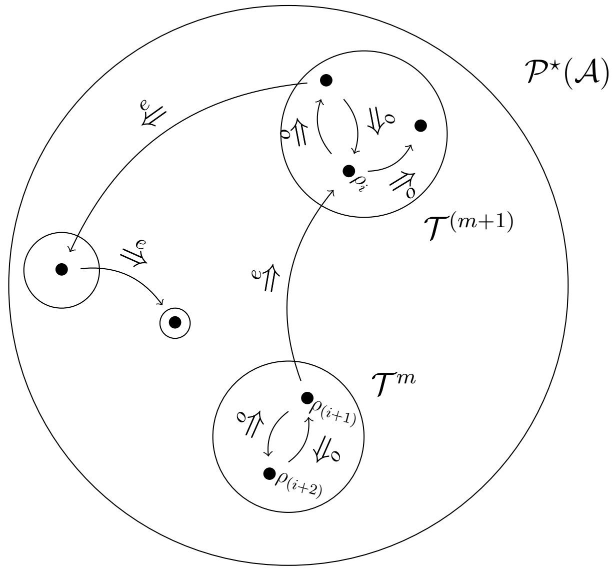


Figure 2: The relationships defined on points of the point space using our two meta-rewrite rules. Though the points are not directly in a term space we illustrate it as if this was the case to represent the observation that points defined over the same term space are associated.

Notice that point spaces are not defined over a single term space but a countably infinite sequence of term spaces. This is necessary for the *ordering* and *expansion* rewrite rules. These rewrite rules allow us to define *curves*.

Definition 9. *A rewrite rule* $\Rightarrow: \mathcal{P}^{\star}(\mathcal{A}) \rightarrow \mathcal{P}^{\star}(\mathcal{A})$ *is* well formed *if for every* $r, d \in \mathcal{P}^{\star}(\mathcal{A})$*, if* $r \Rightarrow d$ *then* $\mathcal{V}(d) \subseteq \mathcal{V}(r)$*.*

There are two important well formed rewrite rules needed for the work outlined in this section. They are as follows:

Definition 10. *We define the well formed rewrite rules* order traverse $\stackrel{o}{\Longrightarrow}: \mathcal{P}^{\star}(\mathcal{A}) \rightarrow \mathcal{P}^{\star}(\mathcal{A})$ *and* expansion $\stackrel{e}{\Longrightarrow}: \mathcal{P}^{\star}(\mathcal{A}) \rightarrow \mathcal{P}^{\star}(\mathcal{A})$ *over* $r, d \in \mathcal{P}^{\star}(\mathcal{A})$*, s.t.* $r = (\rho, v, \{a_1, \cdots a_{\mathcal{A}(\rho,1)}\})$*,* $d = (\rho', v', \{b_1, \cdots b_{\mathcal{A}(\rho,1)}\})$*, as follows:*

- $r \stackrel{o}{\Longrightarrow} d$ *if* $\mathcal{A}(\rho, 0) = \mathcal{A}(\rho', 0)$*, and*
  - $\star$ $\rho' \prec_R \rho$ *and* $v' \preceq_{A(\rho,0)} v$*, or*
  - $\star$ $\rho \preceq_R \rho'$ *and* $v' \prec_{A(\rho,0)} v$

- $r \overset{e}{\Longrightarrow} d$ *if* $\rho' \preceq_R \rho$ *and* $\mathcal{A}(\rho, 0) + 1 = \mathcal{A}(\rho', 0)$

Note that $\overset{o}{\Longrightarrow}$ is invariant the precise relationship between ρ and ρ' in that it only requires them to be ordered by $\preceq_R$ while $\overset{e}{\Longrightarrow}$ requires that ρ' be smaller than ρ over $\preceq_R$. This implies that mutual recursion (i.e. cycling) is allowed but only between path symbols defined over term spaces of the same dimension. This will be essential for the construction of resolution derivations. Also, we would like to note that the condition $\mathcal{A}(\rho, 0) + 1 = \mathcal{A}(\rho', 0)$ can be generalized to $\mathcal{A}(\rho, 0) + w = \mathcal{A}(\rho', 0)$ for $w > 0$ w.l.o.g.

Definition 11. *A substitution space $\mathcal{S}$, is defined as $\mathcal{S} = \{\ \sigma \mid \sigma : \mathcal{P} \to \mathcal{G}\}$*

Subsitution spaces allow us to define *grounded point spaces* and *curves*.

Definition 12. *A* grounded point space *$\mathcal{P}^\star(\mathcal{A}, \sigma)$ is a point space where all of the points are normalized with respect to the substitution σ. A* pre-curve *C is defined over the substitution space with each substitution being identified with a grounded point space $\mathcal{P}^\star(\mathcal{A}, \sigma)$. That is*

$$C = \{A \mid A \subset \mathcal{P}^\star(\mathcal{A}, \sigma)\ ,\ \sigma \in \mathcal{S}\ ,\ |A| \in \mathbb{N}\}$$

A pre-curve is applied to substitutions as one would apply a function to numbers, i.e. $C(\sigma) = A$. A pre-curve C is said to be a curve *if there exists a point $p \in \mathcal{P}^\star(\mathcal{A})$ such that for every substitution σ, $p\sigma \in C(\sigma)$ and for every q, s.t. $q\sigma \in C(\sigma)$ and $p \neq q$, either $p\sigma \overset{o}{\Longrightarrow} q\sigma$ or $p\sigma \overset{e}{\Longrightarrow} q\sigma$. For a curve C fixed to a point p, we write $[C] = p$.*

5.2. Finite Saturated Trees

The next definition introduces a very important type of point set which will be essential to constructing resolution derivations.

Definition 13. *Let C be a curve. Then $\mathcal{BC}(C) = \left\{\sigma \;\middle|\; |C(\sigma)| = 1\ ,\ \sigma \in \mathcal{S}\right\}$ is* the set of basecases of the curve C.

We can further restrict the types of curves we are considering by restricting the variant in the of the set of point sets of a given curve. To define such restrictions we need to define the *set of points of a curve at* σ, that is, $\mathcal{P}(C, \sigma) = \{p \mid p\sigma \in C(\sigma)\}$.

Definition 14. *Let C be a curve over the point space $\mathcal{P}^\star(\mathcal{A})$. We refer to C as* regular *if $\mathcal{P}(C,\mathcal{S}) = \{p \mid p \in \mathcal{P}(C,\sigma)\ ,\ \sigma \in \mathcal{S}\}$ is finite. The set of all regular curves over a point space $\mathcal{P}^\star(\mathcal{A})$ will be denoted by $\mathcal{C}^\star(\mathcal{A})$.*

Notice that for all regular curves there are finitely many ways the rewrite rules $\overset{o}{\Longrightarrow}$ and $\overset{e}{\Longrightarrow}$ can be applied. So far only one point in $\mathcal{P}(C,\mathcal{S})$ is associated with a curve, that is the point $[C]$.

We now define *branches* as regular curves C' of a regular curve C such that there exists σ, θ, $[C']\,\theta \in C(\sigma)$ and $[C']\,\theta \neq [C]\,\sigma$. Here C is the *trunk* that branches attach to at points. A *tree* is a pairing of a trunk and a set of branches where *subtrees* are defined in the standard way. Note that given a trunk C then a curve C' cannot branch from $[C]$. Such a branch would result in an infinite loop. So far this provides an intuitive notion of branches, trunks, and trees. The following definition provides the formal concept.

Definition 15. *A* tree *$\langle C, \mathbf{B}\rangle$ is a pairing of a regular curve C and a set of branches $\mathbf{B}$, where a branch is defined as a pair of a point and a tree. The set of branches attachable to a curve is denoted by $\mathbf{B}^*(C)$, and thus $\mathbf{B} \subset \mathbf{B}^*(C)$.*

$$\mathbf{B}^*(C) = \left\{ (p, \langle C', \mathbf{B}'\rangle) \;\middle|\; \begin{array}{c} C' \in \mathcal{C}^\star(\mathcal{A}, \mathcal{P}_0)\ ,\ \mathbf{B}' \subset \mathbf{B}^*(C')\ , \\ \exists \sigma, \theta \ s.t.\ p \in \mathcal{P}(C,\sigma) \ and\ [C']\,\theta = p\sigma \\ p \neq [C] \end{array} \right\}.$$

Essentially trees can be define recursively as follows:

Definition 16. *The set of all trees $\mathcal{T}^\star(\mathcal{A})$ over $\mathcal{P}^\star(\mathcal{A})$ is defined recursively as follows:*

- *If $C \in \mathcal{C}^\star(\mathcal{A})$ then $\langle C, \emptyset\rangle$ is a tree.*
- *If $\langle C, \mathbf{B}\rangle$ is a tree, $C' \in \mathcal{C}^\star(\mathcal{A})$ and there exists σ, θ such that $q \in \mathcal{P}(C,\sigma)$, $[C']\,\theta = q\sigma$, $q \neq [C]$, and $\mathbf{B}' \subset \mathbf{B}^*(C')$, then*

  $$\langle C, \mathbf{B} \cup (q, \langle C', \mathbf{B}'\rangle)\rangle$$

  is a tree.

Now we can restrict the types of trees we will consider.

Definition 17. *A tree T will be called* finite *if it contains a finite number of curves. A tree T will be referred to as* saturated *if for every curve C in the tree and every $p \in \mathcal{P}(C, \mathcal{S})$ and $p \neq [C]$, there is a curve C' in the tree such that in the subtree of T where C is the trunk the branch set contains (p, T') where C' is the trunk of T'. The set of all* finite saturated trees over an arity function *$\mathcal{A}$ will be denoted by $\mathcal{FS}(\mathcal{T}^{\star})$.*

Definition 18 (Tree Normalization). *Let $T = \langle C, \mathbf{B}\rangle \in \mathcal{FS}(\mathcal{T}^{\star})$, $\sigma \in \mathcal{S}$. We recursively define* tree normalization *$tn(T, \sigma)$ as follows:*

- *Let $C(\sigma) = A$. For each $p\sigma \in A$ such that $(p, \langle C', \mathbf{B}'\rangle) \in \mathbf{B}$ we can compute the substitution θ such that $[C']\,\theta = p$. We now compute $tn(\langle C', \mathbf{B}'\rangle, \theta')$*
- *Let $C(\sigma) = A$. If $\mathbf{B} = \emptyset$ then the recursion terminates.*

Now we can formalize the main theorem of this section.

Theorem 4. *Let $T \in \mathcal{FS}(\mathcal{T}^{\star})$ and $\sigma \in \mathcal{S}$. Then $tn(T, \sigma)$ terminates at the basecases of the curves in T.*

Proof. If normalization does not end at a basecase than there are two possible outcomes, either it terminates at point which is not a basecase or it does not terminate. Terminating at a point which is not a basecase violates Definition 12 concerning the construction of curves. Thus, such a tree is not well-formed. Non-termination also violates Definition 12 because this would imply an infinite descending chain of rewrite applications which is not possible.

Example 3. *To construct a finite saturated tree for the Non-injectivity Assertion schema's refutation we need four path symbols $\{\rho_4, \rho_3, \rho_2, \rho_1\}$ and an arity function $\mathcal{A}$ containing the following mappings, $\mathcal{A}(\rho_4) = (1, 0), \mathcal{A}(\rho_3) = (3, 1), \mathcal{A}(\rho_2) = (3, 1)$, and $\mathcal{A}(\rho_1) = (4, 1)$. The points of the point space $\mathcal{P}^{\star}(\mathcal{A}, \{n\})$ we need are as follows:*

$$
\begin{aligned}
p_1 &= (\rho_4, |n|, \{\}) & p_2 &= (\rho_3, |n, n, n|, \{0\}) \\
p_3 &= (\rho_3, |n, m, k|, \{t\}) & p_4 &= (\rho_3, |n, m, r|, \{0\}) \\
p_5 &= (\rho_2, |n, m, r|, \{w\}) & p_6 &= (\rho_3, |n, s, n|, \{0\}) \\
p_7 &= (\rho_1, |n, m, w, (w \dot{-} r)|, \{r\}) & p_8 &= (\rho_2, |n, m, t|, \{w\}) \\
p_9 &= (\rho_1, |n, s, w, r|, \{t\}) & p_{10} &= (\rho_1, |n, s, w, m|, \{k\})
\end{aligned}
$$

The trunk of the finite saturated tree is the following regular curve

$$C_1 = \{\{p_1, p_2\} \mid \sigma \in \mathcal{S}\}\,.$$

where $[C_1] = p_1$. *We now have the base tree* $\langle C_1, \emptyset\rangle$. *We can now attach a curve* C_2 *as a branch to the tree* $\langle C_1, \emptyset\rangle$ *where* $[C_2] = p_3$. *The definition of the curve* C_2 *is much more complex than the definition of* C_1 *and is as follows*

$$C_2 = \left\{ \begin{array}{c|c} \{p_3, p_4, p_5\} & \sigma \in \Sigma_1 \\ \{p_3, p_5, p_6\} & \sigma \in \Sigma_2 \\ \{p_3, p_5\} & \sigma \in \Sigma_3 \end{array} \right. \qquad \Sigma_2 = \left\{ \sigma \left| \begin{array}{c} \sigma \in \mathcal{S}\ ,\ n\sigma = \overline{n}\ , \\ m\sigma = s(\overline{m})\ ,\ k\sigma = 0\ , \\ t\sigma = \bar{t}\ ,\ r\sigma = 0\ , \\ w\sigma = 0\ ,\ s\sigma = \overline{m} \end{array} \right. \right\}$$

$$\Sigma_1 = \left\{ \sigma \left| \begin{array}{c} \sigma \in \mathcal{S}\ , \\ n\sigma = \overline{n}\ ,\ m\sigma = \overline{m} \\ k\sigma = s(\overline{k})\ ,\ t\sigma = \bar{t} \\ r\sigma = \overline{k}\ ,\ w\sigma = s(\bar{t}) \end{array} \right. \right\} \Sigma_3 = \left\{ \sigma \left| \begin{array}{c} \sigma \in \mathcal{S}\ , \\ n\sigma = \overline{n}\ ,\ m\sigma = 0 \\ k\sigma = 0\ ,\ t\sigma = \bar{t} \\ r\sigma = 0\ ,\ w\sigma = 0 \end{array} \right. \right\}$$

Notice that there are three points in C_2 *upon which we can attach curves. In two cases we attach the curve* C_2 *to itself,* p_4 *and* p_6*, and in the other case we need a new curve* C_3. *So far the tree we have constructed is*

$$\left\langle C_1, \left\{ (p_2, \left\langle C_2, \left\{ \begin{array}{c} (p_4, \langle C_2, \mathbf{B}\rangle), \\ (p_6, \langle C_2, \mathbf{B}\rangle), \\ (p_5, \langle C_3, \emptyset\rangle) \end{array} \right\} \right\rangle \right\} \right\rangle.$$

Notice that for two subtrees we have placed $\mathbf{B}$ *instead of the correct subtrees. The branches attached at this point repeat infinitely often but in a regular fashion. Now we construct* C_3

$$C_3 = \left\{ \begin{array}{c|c} \{p_5, p_7, p_8\} & \sigma \in \Sigma_4 \\ \{p_5, p_6, p_9\} & \sigma \in \Sigma_5 \\ \{p_5, p_9\} & \sigma \in \Sigma_6 \end{array} \right. \qquad \Sigma_5 = \left\{ \sigma \left| \begin{array}{c} \sigma \in \mathcal{S}\ , \\ n\sigma = \overline{n}\ ,\ m\sigma = s(\overline{m})\ , \\ r\sigma = 0\ ,\ t\sigma = 0 \\ w\sigma = \overline{w}\ ,\ s\sigma = \overline{m} \end{array} \right. \right\}$$

$$\Sigma_4 = \left\{ \sigma \left| \begin{array}{c} \sigma \in \mathcal{S}\ , \\ n\sigma = \overline{n}\ ,\ m\sigma = \overline{m} \\ t\sigma = \overline{r}\ ,\ r\sigma = s(\overline{r})\ , \\ w\sigma = \overline{w} \end{array} \right. \right\} \Sigma_6 = \left\{ \sigma \left| \begin{array}{c} \sigma \in \mathcal{S}\ , \\ n\sigma = \overline{n}\ ,\ m\sigma = 0\ , \\ r\sigma = 0\ ,\ t\sigma = 0\ , \\ w\sigma = \overline{w}\ ,\ s\sigma = 0 \end{array} \right. \right\}$$

We attach the curve C_3 to the point p_8, C_2, to point p_6 and a new curve C_4 to points p_7 and p_9. The resulting tree is as follows

$$\left\langle C_1, \left\{ (p_2, \left\langle C_2, \left\{ \begin{array}{c} (p_4, \langle C_2, \mathbf{B} \rangle), \\ (p_6, \langle C_2, \mathbf{B} \rangle), \\ (p_5, \left\langle C_3, \left\{ \begin{array}{c} (p_6, \langle C_2, \mathbf{B} \rangle), \\ (p_7, \langle C_4, \emptyset \rangle), \\ (p_8, \langle C_3, \mathbf{B}' \rangle), \\ (p_9, \langle C_4, \emptyset \rangle) \end{array} \right\} \right\rangle) \end{array} \right\} \right\rangle \right\} \right\rangle .$$

Now we construct the final curve C_4

$$C_4 = \left\{ \begin{array}{c|c} \{p_9, p_{10}\} & \sigma \in \Sigma_7 \\ \{p_9\} & \sigma \in \Sigma_8 \end{array} \right.$$

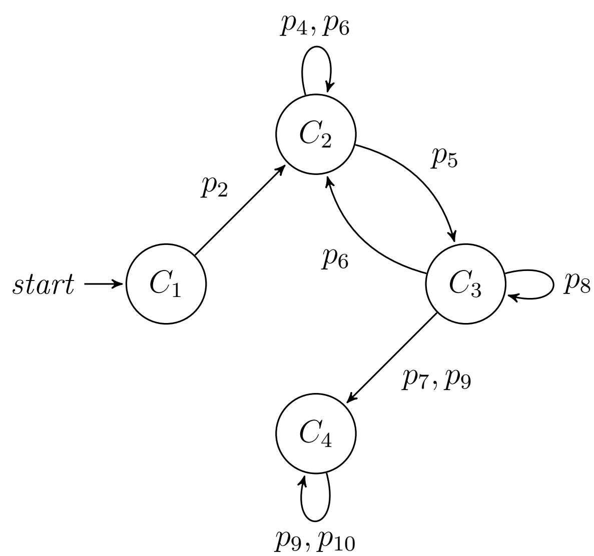
$$\Sigma_7 = \left\{ \sigma \left| \begin{array}{c} \sigma \in \mathcal{S}\ , \\ n\sigma = \overline{n}\ ,\ r\sigma = s(\overline{r})\ ,\ t\sigma = \overline{t} \\ w\sigma = \overline{w}\ ,\ s\sigma = \overline{s}\ ,\ m\sigma = \overline{r} \\ ,\ k\sigma = s(\overline{t}) \end{array} \right. \right\}$$

$$\Sigma_8 = \left\{ \sigma \left| \begin{array}{c} \sigma \in \mathcal{S}\ , \\ n\sigma = \overline{n}\ ,\ s\sigma = \overline{s}\ , \\ r\sigma = 0\ ,\ t\sigma = \overline{t}\ , \\ w\sigma = \overline{w} \end{array} \right. \right\}$$

The final tree is

$$\left\langle C_1, \left\{ \left(p_2, \left\langle C_2, \left\{ \begin{array}{c} (p_4, \langle C_2, \mathbf{B}\rangle), \\ (p_6, \langle C_2, \mathbf{B}\rangle), \\ \left(p_5, \left\langle C_3, \left\{ \begin{array}{c} (p_6, \langle C_2, \mathbf{B}\rangle), \\ (p_7, \langle C_4, \{(p_{10}, \langle C_4, \mathbf{B}''\rangle)\}\rangle), \\ (p_8, \langle C_3, \mathbf{B}'\rangle), \\ (p_9, \langle C_4, \{(p_{10}, \langle C_4, \mathbf{B}''\rangle)\}\rangle) \end{array} \right\} \right\rangle \right) \end{array} \right\} \right\rangle \right) \right\} \right\rangle.$$

Notice that the tree is finite and saturated. Below is an automaton representation of the recursive tree.



In the next section, we define resolution derivations and decorations of finite saturated trees

5.3. Resolution Decorations of Finite Saturated Trees

In order to properly use the finite saturated trees introduced in the previous section we need to not only decorate the point sets but the points themselves as well. The necessity of such point decoration comes from the fact that a resolution derivation might not only need a "scratch pad" for the numeric terms, but also one for the individual terms. So, far we have not consider this issue and will address it in this section.

Definition 19. *A* model *is a set of non-contradictory literals* $\mathbf{M}$. *We allow so called* model variables *within our resolution derivations.*

Definition 20 (Point Decoration). *Let* $p \in \mathcal{P}^{\star}(\mathcal{A})$ *and* $t_1, \cdots, t_k$ *be terms of the individual sort and* $\mathbf{M}$ *be a model. A point Decoration of* p *is a triple* $(p, t_1, \cdots, t_k, \mathbf{M})$.

Definition 21 (Resolution Derivation). *A* resolution derivation *can be built recursively as follows:*

- *A point decoration is a resolution derivation.*
- *A clause* C *composed with a model or a model variable is a derivation.*
- *Given resolution derivations* R_1 *and* R_2 *and a substitution* σ

$$\frac{R_1 \qquad R_2}{es(R_1) \setminus \{P\} \circ es(R_2) \setminus \{\neg P\}}\, res(\sigma)$$

  is a resolution derivation if $Q\sigma = P$ *and* $E\sigma = \neg P$ *for literal* Q *in* R_1 *and* E *in* R_2.

Definition 22 (Decoration of a Point Set). *Let* $C \in \mathcal{C}^{\star}(\mathcal{A})$, σ *a substitution, and* R *a resolution derivation such that for each point in* $q\sigma \in C(\sigma)$ *such that* $q \neq [C]$ *there is a point decoration in* R *matching* q *and* $\mathcal{V}(R) \subseteq \mathcal{V}(\mathcal{P}(C, \sigma))$ *. Then a* decoration of $C(\sigma)$ by R *is a pair* (σ, R). *If more then one substitution is decorated by* R, *then we can write* (S, R) *where* S *is a set of substitutions.*

Definition 23 (Decoration of a Tree). *Let* $T \in \mathcal{FS}(\mathcal{T}^{\star})$ *and* $\mathbf{RD}$ *a set of resolution derivations. A* decoration of T by $\mathbf{RD}$, $dec(RD, T)$ *is a set of distinct triples* (C, σ, R), *where* $C \in T$, σ *is a substitution, and* $R \in RD$ *such that* (σ, R) *is a decoration of the point set* $C(\sigma)$. *If for every* $C \in T$,

$$\mathcal{S} \equiv \bigcup_{\substack{(C', \sigma, R) \in dec(RD, T) \\ C' = C}} \sigma$$

then we refer to the decoration as festive.

From now on we will only consider *festive decorations* of finite saturated trees. The following section presents a festive decoration of the tree constructed in the previous section. This festive decoration is the refutation of the characteristic formula belonging to the Non-Injectivity Assertion schema which we discussed earlier.

6. Analysis of the Non-injectivity Assertion Schema

We now present a decoration of the finite saturated tree we constructed in the previous section. This decoration formalizes the refutation of the characteristic formula belonging to the Non-Injectivity Assertion schema which we discussed earlier.

The decoration of the point sets of the curve C_1 is as follows for all substitutions in $\mathcal{S}$.

$$\Big(C_1, \mathcal{S}, \left(\left(\rho_3, |n,n,n|, \{0\}\right), X, \vdash\right)\Big)$$

The decoration of the point sets of the curve C_2 is as follows:
$\Big(C_2, \Sigma_1, R_1\Big)$ where R_1 is

$$\frac{\begin{array}{c}((\rho_2, |n,m,r|, \{w\}), \\ \{f(x) = m \vdash \circ\mathcal{M}\})\end{array} \qquad \begin{array}{c}((\rho_3, |n,m,r|, \{0\}), \\ \{\vdash f(x) = m \circ \mathcal{M}\})\end{array}}{\mathcal{M}}\ \mathrm{res}(\sigma)$$

where $\sigma = \{x \leftarrow m(r,0)\}$. Next consider $\Big(C_2, \Sigma_2, R_2\Big)$ where R_2 is

$$\frac{\begin{array}{c}((\rho_2, |n,m,r|, \\ \{w\}), \{f(x) = m \vdash \circ\mathcal{C}\})\end{array} \qquad \begin{array}{c}((\rho_3, |n,s,n|, \{0\}), \\ \{\vdash f(x) = m \circ \mathcal{C}\})\end{array}}{\mathcal{M}}\ \mathrm{res}(\sigma)$$

where $\sigma = \{x \leftarrow m(r,0)\}$. Next consider $\Big(C_2, \Sigma_3, R_3\Big)$ where R_3 is

$$\frac{\begin{array}{c}(\rho_2, |n,m,r|, \{w\}), \\ \{f(x) = m \vdash \circ\mathcal{M}\})\end{array} \qquad \vdash f(X) = m \circ \mathcal{M}}{\mathcal{M}}\ \mathrm{res}(\sigma)$$

where $\sigma = \{x \leftarrow m(r,0)\}$. The decoration of the point sets of the curve C_3 is as follows: $\Big(C_3, \Sigma_4, R_4\Big)$ where R_4 is

$$\frac{\begin{array}{c}((\rho_1, |n,m,w,(w \dot{-} r)|, \{r\}) \\ \{f(x) = m \vdash s(x) \leq y \circ \mathcal{M}\})\end{array} \qquad s(x) \leq y \vdash \circ\mathcal{M}}{\vdots}\ \mathrm{res}(\sigma)$$

$$\dfrac{\begin{matrix}\vdots \\ \mathcal{M}\end{matrix} \qquad ((\rho_2, |n, m, t|, \{w\}), \{\vdash f(x) = m \circ \mathcal{M}\}}{\mathcal{M}} \operatorname{res}(\theta)$$

where $\sigma = \left\{ \ x \leftarrow m(r, 0), \ \ y \leftarrow m(w, 0) \ \right\}$ and $\theta = \left\{ \ x \leftarrow m(r, 0) \ \right\}$. Next consider $\left(C_3, \Sigma_5, R_5\right)$ where R_5 is

$$\dfrac{\begin{matrix}(\rho_1, |n, s, w, r|, \{t\}), \\ \{f(x) = m \vdash s(x) \leq y \circ \mathcal{M}\})\end{matrix} \qquad s(x) \leq y \vdash \circ\mathcal{M}}{\vdots} \operatorname{res}(\sigma)$$

$$\dfrac{\begin{matrix}\vdots \\ \mathcal{M}\end{matrix} \qquad \rho_3(|n, s, n|, \{0\}, \{\vdash f(x) = m \circ \mathcal{M}\})}{\mathcal{M}} \operatorname{res}(\theta)$$

where $\sigma = \left\{ \ x \leftarrow m(r, 0), y \leftarrow m(w, 0) \ \right\}$ and $\theta = \left\{ \ x \leftarrow m(0, 0) \ \right\}$. Next consider $\left(C_3, \Sigma_5, R_6\right)$ where R_6 is

$$\dfrac{\begin{matrix}\rho_1(|n, s, m, w|, \{k\}, \\ \{f(x) = m \vdash s(x) \leq y \circ \mathcal{M}\})\end{matrix} \qquad s(x) \leq y \vdash \circ\mathcal{M}}{\vdots} \operatorname{res}(\sigma)$$

$$\dfrac{\begin{matrix}\vdots \\ \mathcal{M}\end{matrix} \qquad \vdash f(x) = m \circ \mathcal{M}}{\mathcal{M}} \operatorname{res}\left(\left\{ \ x \leftarrow m(r, 0) \ \right\}\right)$$

where $\sigma = \left\{ \ x \leftarrow m(r, 0), \ \ y \leftarrow m(w, 0) \ \right\}$. And finally, the decoration of the point sets of the curve C_4 is as follows:
$\left(C_4, \Sigma_7, R_7\right)$ where R_7 is

$$\dfrac{\begin{matrix}((\rho_1, |n, s, w, r|, \{t\}), \\ \{\vdash max(s(x), x) \leq y \circ \mathcal{M}\})\end{matrix} \qquad max(s(x), x) \leq x \vdash \circ\mathcal{M}}{\mathcal{M}} \operatorname{res}(\sigma)$$

where $\sigma = \left\{ \ x \leftarrow m(t, 0), \ \ y \leftarrow m(w, 0) \ \right\}$. Next consider $\left(C_4, \Sigma_8, R_8\right)$ where R_8 is

$$\frac{\vdash max(s(x),x) \leq y \circ \mathcal{M} \qquad max(s(x),x) \leq y \vdash \circ\mathcal{M}}{\mathcal{M}} \text{res}(\sigma)$$

where $\sigma = \left\{ \ x \leftarrow m(t,0), y \leftarrow m(w,0) \ \right\}$. Let $\mathbf{RD} = \{((\rho_3, |n,n,n|, \{0\}), \vdash)\} \cup \{R_i\}_{i=1}^{8}$ and T the finite saturated tree constructed in the previous section, then

$$dec(T, \mathbf{RD}) = \left\{ \begin{array}{cc} \Big(C_1, \mathcal{S}, ((\rho_3, |n,n,n|, \{0\}), \vdash)\Big) & \Big(C_2, \Sigma_1, R_1\Big) \\ \Big(C_2, \Sigma_2, R_2\Big) & \Big(C_2, \Sigma_3, R_3\Big) \\ \Big(C_3, \Sigma_4, R_4\Big) & \Big(C_3, \Sigma_5, R_5\Big) \\ \Big(C_3, \Sigma_6, R_6\Big) & \Big(C_4, \Sigma_7, R_7\Big) \\ \Big(C_4, \Sigma_8, R_8\Big) & \end{array} \right.$$

is a festive decoration of T. Any point set not referenced in $dec(T, \mathbf{RD})$ is decorated by the empty clause because it is not reachable by tree normalization.

7. Conclusion

In this work we have presented a mathematically interesting proof schema as well as an informal proof of its end sequent. A characteristic formula is then extracted from this proof schema and a informal proof of the schema of refutations is then constructed. This schema of refutations has been know for quite some time [5], but none of the existing formal systems for schematic proof analysis can express this schema formally. In order to tackle this problem we introduce a recursive tree construction which is modular in the sense that the calculus is defined separately from the recursive construction. Furthermore the recursion formalized by our tree constructions is more expressive than previous formalizations [12, 15]. We then show how to decorate these trees with resolution derivations and provide a formal representation of the Non-injectivity Assertion schema's characteristic formula's refutation. There are still many questions which have to be asked concerning this new formalism, for example how expressive is it? Also, is it complete for a significant fragment of proof schemata, possibly a fragment of P-schema [10]. Moreover, we have yet to extend Herbrand's theorem to refutations defined over finite saturated trees [12, 15]. While it would not be possible to extend the method of [12, 15] to finite saturated trees in such a way that Herbrand's theorem holds, the novel concept of *generalization variables* introduced in [7] perfectly fits the introduced formalization in that it is also a modular method of representing a schema of substitutions. Extension of generalization variables to festive decorations of finite saturated trees is the goal of future work.

Acknowledgements: We thank Michael Lettmann and Alexander Leitsch for contributing to the polishing and development of this work.

References

[1] Aravantinos, V., Echenim, M., Peltier, N., Apr. 2013. A resolution calculus for first-order schemata. Fundam. Inf. 125 (2), 101–133.
URL `http://dx.doi.org/10.3233/FI-2013-855`

[2] Baaz, M., Hetzl, S., Leitsch, A., Richter, C., Spohr, H., Aug. 2008. Ceres: An analysis of Fürstenberg's proof of the infinity of primes. Theoretical Computer Science 403 (2-3), 160–175.

[3] Baaz, M., Leitsch, A., 2000. Cut-elimination and redundancy-elimination by resolution. Journal of Symbolic Computation 29, 149–176.

[4] Brotherston, J., 2005. Cyclic proofs for first-order logic with inductive definitions. In: Tableaux'05. Vol. 3702 of Lecture Notes in Comp. Sci. pp. 78–92.

[5] Cerna, D. M., 2015. Advances in schematic cut elimination. Ph.D. thesis, Technical University of Vienna, http://media.obvsg.at/p-AC12246421-2001.

[6] Cerna, D. M., Leitsch, A., 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. pp. 241–256.

[7] Cerna, D. M., Leitsch, A., Lolic, A., Jan. 2018. A Resolution Calculus for Recursive Clause Sets [Preprint]. Risc report series, Research Institute for Symbolic Computation (RISC), Johannes Kepler University Linz, Schloss Hagenberg, 4232 Hagenberg, Austria, in review.

[8] Cerna, D. M., Lettmann, M., 2017. Integrating a global induction mechanism into a sequent calculus. In: Schmidt, R. A., Nalon, C. (Eds.), Automated Reasoning with Analytic Tableaux and Related Methods. Springer International Publishing, Cham, pp. 278–294.

[9] Cerna, D. M., Lettmann, M., Sept. 2017. Towards a clausal analysis of proof schemata. In: SYNASC'17. IEEE Xplorer. IEEE.

[10] Cerna, D. M., Lolic, A., January 2018. Proof Schemata for Theories equivalent to PA: on the Benefit of Conservative Reflection Principles. Tech. rep., RISC, in review.
URL `https://arxiv.org/abs/1711.10994`

[11] Coq development team, T., 2009. The Coq proof assistant reference manual. LogiCal Project.
URL `http://coq.inria.fr/doc/`

[12] Dunchev, C., Leitsch, A., Rukhaia, M., Weller, D., 2013. Cut-elimination and proof schemata. In: Logic, Language, and Computation. pp. 117–136.

[13] Gentzen, G., Dec. 1935. Untersuchungen über das logische Schließen I. Mathematische Zeitschrift 39 (1), 176–210.

[14] Kovcs, L., Voronkov, A., 2013. First-order theorem proving and vampire. In: Sharygina, N., Veith, H. (Eds.), Computer Aided Verification. Vol. 8044 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 1–35.

[15] Leitsch, A., Peltier, N., Weller, D., 2017. CERES for first-order schemata. J. Log. Comput. 27 (7), 1897–1954.
URL `https://doi.org/10.1093/logcom/exx003`

[16] Mendelson, E., 2009. Introduction to Mathematical Logic, 5th Edition. Chapman & Hall/CRC.

[17] Simmons, H., Sep 1988. The realm of primitive recursion. Archive for Mathematical Logic 27 (2), 177–188.
URL `https://doi.org/10.1007/BF01620765`

[18] Sutcliffe, G., 2010. The tptp world - infrastructure for automated reasoning. In: Proceedings of the 16th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning. LPAR'10. Springer-Verlag, pp. 1–12.

[19] Takeuti, G., 1975. Proof Theory. Vol. 81 of Studies in logic and the foundations of mathematics. American Elsevier Pub.

[20] Urban, C., 2000. Classical logic and computation. Ph.D. thesis, University of Cambridge.

[21] Weidenbach, C., Dimova, D., Fietzke, A., Kumar, R., Suda, M., Wischnewski, P., 2009. Spass version 3.5. In: Proceedings of the 22Nd International Conference on Automated Deduction. CADE-22. Springer-Verlag, Berlin, Heidelberg, pp. 140–145.