Fundamenta Informaticae XX (2018) 1–27
DOI 10.3233/FI-2016-0000
IOS Press

An Incompleteness Result for Loop Discovery Based Recursive Provers: [Preprint]

David M. Cerna*
Research Institute for Symbolic Computation (RISC)
Johannes Kepler University
Linz, Austria
David.Cerna@risc.jku.at

Abstract. Loop based tableau and resolution provers for recursively defined formula and clause sets have been thoroughly investigated by Aravantinos *et al.* culminating in their work "A Resolution Calculus for First-order Schemata". The goal of these methods is to transform the given formula or clause set into a state which is invariant a shift of the free parameter, i.e. n shifted to $n-1$. By invariant, we mean a shift in the value of the free parameter preserving satisfiability. Recent work by Leitsch *et al.* "CERES for First-Order Schemata" used a looping resolution prover to transform recursively defined formal proofs into a normal form accepting "Herbrand sequent" extraction. While such looping provers can handle expressive classes of recursively defined formulas we show that a quite simple recursive formula resulting from proof transformation can have invariants which cannot be discovered by the discussed loop discovery method. The issue being that any substitution grounding the invariant set of clauses is itself dependent on the free parameter and needs to be shifted as well. This points to a limitation of such loop discovery methods given that quite weak mathematically relevant statements can easily overpower the mechanism.

1. Introduction

Proof schemata are recursively defined infinite sequences of finite LK-proofs indexed by a single of free numeric parameter. When grounded (with respect to the free parameter) and

*This research is supported by FWF project P28789-N32

normalized (similar to primitive recursive definitions) the result is an *LK-proof*. Schematic CERES [1] was developed in order to generalize Herbrand's Theorem [2] to fragments of arithmetic presentable as proof schemata, essentially by replacing induction of Gentzen's **LK**-calculus for PA by proof calls. One can think of this calculus as similar to cyclic proofs [3, 4] where the cycling is explicitly defined as a collection of primitive recursive definitions. This means that a global trace condition is not necessary, i.e. there are local conditions which can be used for proof construction [5]. Development of schematic CERES was inspired by the analysis of Fürstenberg's proof of the infinitude of primes by Baaz et al. [6]. However, unlike the first-order version of CERES [7] our understanding of theorem proving for recursively defined first order formulas is nowhere close to as deep as our understanding of theorem proving for first order formula. Powerful theorem proving methods for expressive fragments of inductive theories has been and is currently of great interest to the theorem proving community [8, 9, 10, 11, 12, 13, 14, 15, 16].

The lack of powerful theorem provers for induction illustrates the difficulty of the problem and points towards the apparent weakness of schematic CERES when compared to the first-order variant [7]. Needless to say, this necessitates an interactive, over an automated use. For example, the analysis of a relatively weak version of the infinitary pigeonhole principle [17] required heavy interactive use of state of the art theorem provers [18, 19] to generate refutations of instances directing us towards an induction invariant. Recently, the earlier schematic CERES method was integrated with the superposition prover of Aravantinos *et al.* for recursively defined clause sets based on loop discovery [20]. A similar tableau prover was developed for propositional schema in earlier work [21, 22]. The resulting method is cut-elimination complete with respect to the refutational power of the superposition prover. But how expressive is the prover?

A detailed analysis answering this question remains to be done, however in this work we provide a recursively defined first-order formula which cannot be refuted using the loop discovery methods of [23], thus providing an upper bound. The problem being that their loop discovery method cannot find nested loops which are interdependent, i.e. the invariant is a loop itself, or in other words enforcing a loop based invariant necessitates a secondary invariant. This is not the case for inductive theorem provers based on other invariant finding mechanisms such as tree grammar based methods [10] which are able to find a solution. Thus, the Σ_1-incompleteness discussed in [24] is a corollary of our construction. Furthermore, this points to an interesting use of the schematic method, that is, as a formalism for bettering our understanding of inductive theorem proving through formal proof transformation. For example, the inductive definitions resulting from analysis of proof schemata are in many cases of mathematical interest and can thus provide key examples to the community for libraries like TPTP [25] and TIP [26] of which the author has already contributed.

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 formalization of the 1-strict decreasing monotone schema using a sequence of Δ_1 cuts. From this proof schema we extract an inductive definition

of the cut structure C.

- In Section 6, we provide a refutation of the inductive definition extracted in Section 4 which we prove to be unique. The uniqueness (modulo renaming) of this refutation implies that any inductive theorem prover which will attempt to refute this problem is required to find this particular refutation.
- In Section 7, we discuss the implications of the previous sections and future research directions.

2. Preliminaries

In this section we introduce the the first-order sequent calculus and a schematic formalism extending it. The proofs of this extension are referred to as *proof schema* and can be thought of as connected graphs of proofs attached to each other by *links* between initial sequents and end sequents. we then introduce some basic proof transformation techniques such as *characteristic formula extraction*. As discussed in [20], such characteristic formula can be translated into a schematic clause set *á la* Aravantinos *et al.* which can be refuted using the superposition prover defined in [23]. This calculus uses a loop finding mechanisms which we show is incomplete by providing a characteristic formula which can only be refuted by discovering an invariant which is loop_2.

2.1. The First-Order LK-calculus

We discuss the term language of our logic in the next section concerning proof schema given that there are additional requirements beyond the term construction of first-order logic. For now 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 [27, 28]. Formulas are built inductively as usual from a countable set $\mathcal{P}$ of predicate atoms using the logical connectives $\neg, \wedge, \vee, \rightarrow, \forall, \exists$.

Construction of formal proofs will be performed using the LK-calculus [29]. This calculus will provide the foundation of the LKS-calculus which extends the LK-calculus by *proof links* and equational rules for inductive definitions. The LK-calculus is defined for *sequents* which are pairs of multisets of formulas $\Pi \vdash \Delta$ which can be interpreted as a 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 referred to as the *end sequent*, s.t. any two adjacent sequents are part of a sound application of one of the inference rules of Figure 1. The lower sequent of an inference rule is referred to as the *main sequent* and the upper 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*.

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

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

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

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

$$\frac{\Gamma \vdash \Delta, C \quad \Gamma \vdash \Delta, D}{\Gamma \vdash \Delta, C \wedge D}\,\wedge\mathrm{:r}$$

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

$$\frac{C, \Gamma \vdash \Delta \quad D, \Gamma \vdash \Delta}{C \vee D, \Gamma \vdash \Delta}\,\vee\mathrm{:l}$$

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

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

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

$$\frac{\Gamma \vdash \Delta, C \quad C, \Gamma \vdash \Delta}{\Gamma \vdash \Delta}\,\mathrm{cut}$$

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

2.2. The Schematic Language

We us a two-sorted term language, the individual sort ι, and the *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 only need proof schema indexed by a single parameter. For information concerning proof schema with multiple parameters and its relationship to Peano arithmetic please see [30]. 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.

We extend our first-order language by *defined function symbols*. We refer to this extension as the *schematic first-order language* which allows an (infinite) sequence of 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 formula containing defined predicate symbols are also formula schemata.

We denote defined symbols by $\hat{}$, 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 $\mathcal{E}$ 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 formula schema. The $\mathcal{E}$-rule is reversible.

Example 2.1. 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 generalize sequent to *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 2.2. (LKE)
Let $\mathcal{E}$ be an equational theory consisting of inductive definitions. 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

$$\frac{\overset{\cdots}{(\varphi, \overline{t})}}{S(\overline{t})}$$

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 an ordered sequence of *proof schema components* which allow restricted use 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 [1, 20]). For those interested in the multiple parameter case, see [30].

Definition 3.1. (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 3.2. 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:

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

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 3.3. (Proof schema [1])
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 3.4. (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:

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

ν is as follows:

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

where

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

and ν_2 is

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

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 detail in [20], here we cover the concepts necessary for comprehension of the rest of this paper. Evaluation is defined as follows:

Definition 3.5. 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 $C^1.1 \Downarrow_\gamma$.

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

Lemma 3.6. 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 3.6 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 within the LK-caculus, i.e. no links or $\mathcal{E}$ rules. Thus, this statement can be strengthen to the following theorem.

Theorem 3.7. 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 now move on to the extraction of the characteristic formula schema from a given proof schema, an essential concept for constructing the inductive definition need for our main result.

3.1. Characteristic Formula Extraction

Let Φ be skolemized proof schema[1] containing cuts inferences, then extraction of a *schematic characteristic formula* representing cut structure of Φ is possible. The characteristic formula

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

is an integral part of both the CERES method and schematic CERES method as presented in [20]. A 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 the proof schema. One can construct a schematic version of the characteristic formula by accounting for formula occurrences of previous recursive calls. In some cases these occurrences are also cut ancestors. To capture the cut-status of formula occurrences through proof links, i.e. through recursive calls, cut ancestor status becomes dependent on *configurations* of formula occurrences which are treated similarly to standard cut ancestors. When necessary we denote the current configuration by Ω.

Similar to proof schema evaluation we introduce so called *characteristic formula symbols* which are dependent on the 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 symbols 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 3.8. 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 $\Theta\rho(\pi, \Omega) = \mathcal{F}(\Xi(A))$[2]
- 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

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

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 *1-Strict Monotone Assertion Schema*.

4. The 1-Strict Monotone Assertion (1-SMA)

Consider a total function $f : \mathbb{N} \to \mathbb{N}$. we refer to such f as *k-strict monotone decreasing* if there exists a set of $A \subset \mathbb{N}$, whose cardinality is k, s.t. if $x \in A$ then $f(x) = f(x+1)$, and if $x \in \mathbb{N} \setminus A$ then $f(x) < f(x+1)$. The 1-strict Monotone Assertion concerns a total function with a range restricted to a the finite set $\{0, \cdots, n\}$. The statement is as follows: for every $n \geq 0$, and total monotone decreasing function $f : \mathbb{N} \to \{0, \cdots, n\}$, f is at least 1-strict monotone decreasing. This is actually a quite weak statement being that it is obviously the case that f is at least k-strict monotone decreasing for every k, but for our work this condition is enough.

In this section we define a formal proof schema for 1-SMA with a sequence of Δ_1 cuts. From this proof schema we extract a characteristic formula which is the object of our analysis. we will show that there is a unique sequence of refutations of this characteristic formula. Furthermore, The extracted characteristic formula can be transformed into a clause set whose invariant is beyond the loop detecting mechanism of [21, 23]. This highlights an essential problem with the loop detection method, i.e. even extremely simple examples can require more powerful loop discovery methods then those which have been formalized so far. Note that 1-SMA is simple but mathematically relevant, it is essentially a variant of the infinitary pigeonhole principle.

Our formalized proof of 1-SMA represents the total monotone decreasing function $f : \mathbb{N} \to \{0, \cdots, n\}$ as a function from ι to ω, $f(\cdot) : \iota \to \omega$. Essentially, ω plays the role of the finite set. We also define two predicate symbols over ω, $=: \omega \to \omega \to o$ and $<: \omega \to \omega \to o$ whose properties are defined by the axioms of Definition 4.1. We define a successor and least element over ι to represent the range of f , namely, $\mathbf{S}(\cdot) : \iota \to \iota$ and a constant $\mathbf{0} : \iota$. We mark the individual sort successor and zero with bold to differentiate them from the numeric sort counterparts. Our formalization uses the following theory:

Definition 4.1. (1-strict monotone decreasing theory)

$$\left(\bigvee_{i=0}^{k} i = f(x)\right) \to f(x) < s(k) \tag{1}$$

$$\left(\bigvee_{i=0}^{k} i = f(\mathbf{s}(x))\right) \to f(x) < s(k) \tag{2}$$

$$(k = f(x) \wedge k = f(\mathbf{s}(x))) \to f(x) = f(\mathbf{s}(x)) \tag{3}$$

$$\neg f(\mathbf{0}) < 0 \tag{4}$$

$$f(\mathbf{s}(x)) < s(k) \to (k = f(\mathbf{s}(x)) \vee f(x) < k) \tag{5}$$

$$f(x) < s(k) \to (k = f(x) \vee f(x) < k) \tag{6}$$

Note that the first two statements of the theory contain iterations. This is not typically allowed. One should interpret these iterations as abbreviations of the following schema of axioms:

$$0 = f(x) \vdash f(x) < s(k), \qquad 1 = f(x) \vdash f(x) < s(k), \quad \cdots \qquad k = f(x) \vdash f(x) < s(k)$$

Any axiom containing k is an axiom schema where k can range over the numeric sort. These statements can either be added to the context of the sequents or, by replacing the $\to$ by the sequent symbol, considered as an axiomatic extension of the **LK**-calculus.

The end-sequent and the sequence of cut formulas are as follows:

Definition 4.2. (End-sequent 1-SMA)

$$\forall x \bigvee_{i=0}^{n} i = f(x) \vdash \exists x(f(x) = f(\mathbf{s}(x)))$$

where n ranges over the numeric sort.

Note that we do not include our theory in the context of our end-sequent, rather we use the theory as initial sequents.

Definition 4.3. (Sequence of cut formulas)

$$\exists x(n = f(x) \wedge n = f(\mathbf{S}(x))) \vee \forall x(f(x) < n)$$

where n ranges over the numeric sort.

The proof schema formalizing the 1-SMA schema consists of two proof components $\langle(\psi, \pi, \nu), (\varphi, \pi', \nu')\rangle$ where the individual components are defined as follows:

Proof: π

$$\dfrac{\dfrac{\dfrac{\dfrac{0 = f(\alpha), 0 = f(\mathbf{S}(\alpha)) \vdash}{f(\alpha) = f(\mathbf{S}(\alpha))}}{\exists x(0 = f(x) \wedge 0 = f(\mathbf{S}(x))) \vdash}\textit{unary rules} \quad \dfrac{f(\mathbf{0}) < 0 \vdash}{\forall x(f(x) < 0) \vdash}\forall : l}{\exists x\,(f(x) = f(\mathbf{S}(x)))}\wedge : r}{\exists x(0 = f(x) \wedge 0 = f(\mathbf{S}(x))) \vee \forall x(f(x) < 0) \vdash \exists x\,(f(x) = f(\mathbf{S}(x)))}$$

$$(A)$$

$$\dfrac{\dfrac{\dfrac{0 = f(\mathbf{S}(\mathbf{0})) \vdash 0 = f(\mathbf{S}(\mathbf{0})) \qquad 0 = f(\mathbf{0}) \vdash 0 = f(\mathbf{0})}{0 = f(\mathbf{0}), 0 = f(\mathbf{S}(\mathbf{0})) \vdash 0 = f(\mathbf{0}) \wedge 0 = f(\mathbf{S}(\mathbf{0})))}\wedge : r}{\forall x \left(\bigvee_{i=0}^{0} i = f(x)\right) \vdash}\textit{unary rules} \quad (A)}{\forall x \left(\bigvee_{i=0}^{0} i = f(x)\right) \vdash \exists x\,(f(x) = f(\mathbf{S}(x)))}\textit{cut}$$

Proof: ν

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{n+1 = f(\alpha), \quad n+1 = f(\mathbf{S}(\alpha)) \vdash} \quad \dfrac{\bigvee_{i=0}^{n} i = f(\mathbf{S}(\alpha)) \vdash}{f(\mathbf{S}(\alpha)) < n+1}}{n+1 = f(\alpha) \wedge n+1 = f(\mathbf{S}(\alpha)))}\vee : l}{n+1 = f(\alpha), \quad \bigvee_{i=0}^{n+1} i = f(\mathbf{S}(\alpha)) \vdash \quad n+1 = f(\alpha) \wedge n+1 = f(\mathbf{S}(\alpha))), \quad f(\mathbf{S}(\alpha)) < n+1} \quad \dfrac{\bigvee_{i=0}^{n} i = f(\alpha) \vdash}{f(\alpha) < n+1}\wedge : r}{\bigvee_{i=0}^{n+1} i = f(\alpha), \quad \bigvee_{i=0}^{n+1} i = f(\mathbf{S}(\alpha)) \vdash \quad n+1 = f(\alpha) \wedge n+1 = f(\mathbf{S}(\alpha))), \quad f(\mathbf{S}(\alpha)) < n+1}\textit{unary rules}}{\forall x \left(\bigvee_{i=0}^{n+1} i = f(x)\right) \vdash}$$

$$\exists x(n+1 = f(x) \wedge n+1 = f(\mathbf{S}(x))) \vee \forall x(f(x) < n+1))$$

$$(A)$$

$$\dfrac{\dfrac{\varphi(n+1)}{\exists x(n+1 = f(x) \wedge n+1 = f(\mathbf{S}(x))) \vee \forall x(f(x) < n+1) \vdash \exists x\,(f(x) = f(\mathbf{S}(x)))}\,(\mathrm{A}) \qquad \exists x\,(f(x) = f(\mathbf{S}(x)))}{\forall x \left(\bigvee_{i=0}^{n+1} i = f(x)\right) \vdash \exists x\,(f(x) = f(\mathbf{S}(x)))}\,cut$$

Proof: π'

$$\dfrac{\dfrac{\dfrac{0 = f(\alpha), 0 = f(\mathbf{S}(\alpha)) \vdash f(\alpha) = f(\mathbf{S}(\alpha))}{\dfrac{\textit{unary rules}}{\exists x(0 = f(x) \wedge 0 = f(\mathbf{S}(x))) \vdash \exists x\,(f(x) = f(\mathbf{S}(x)))}} \qquad \dfrac{f(\mathbf{0}) < 0 \vdash}{\forall x(f(x) < 0) \vdash}\,\forall\text{:r}}{\exists x(0 = f(x) \wedge 0 = f(\mathbf{S}(x))) \vee \forall x(f(x) < 0) \vdash \exists x\,(f(x) = f(\mathbf{S}(x)))}\,\vee\text{:l}$$

Proof: ν'

$$\dfrac{\dfrac{\dfrac{n+1 = f(\alpha), n+1 = f(\mathbf{S}(\alpha))) \vdash f(\alpha) = f(\mathbf{S}(\alpha))}{\textit{unary rules}}}{\exists x(n+1 = f(x) \wedge n+1 = f(\mathbf{S}(x))) \vdash \exists x\,(f(x) = f(\mathbf{S}(x)))} \qquad \dfrac{\dfrac{\vdots}{f(\mathbf{S}(\alpha)) < n+1), f(\alpha) < n+1) \vdash n = f(\alpha) \wedge n = f(\mathbf{S}(\alpha))), f(\alpha) < n}}{\dfrac{\textit{unary rules}}{\forall x(f(x) < n+1) \vdash \exists x(n = f(x) \wedge n = f(\mathbf{S}(x))), \forall x(f(x) < n)}}\,\vee\text{:l}}{\dfrac{\exists x(n+1 = f(x) \wedge n+1 = f(\mathbf{S}(x))) \vee \forall x(f(x) < n+1) \vdash \exists x(n = f(x) \wedge n = f(\mathbf{S}(x))), \forall x(f(x) < n), \exists x\,(f(x) = f(\mathbf{S}(x)))}{\exists x(n+1 = f(x) \wedge n+1 = f(\mathbf{S}(x))) \vee \forall x(f(x) < n+1) \vdash \exists x(n = f(x) \wedge n = f(\mathbf{S}(x))) \vee \forall x(f(x) < n), \exists x\,(f(x) = f(\mathbf{S}(x)))}\,\vee\text{:r}}$$

(A)

$$\dfrac{\varphi(n)}{\exists x(n = f(x) \wedge n = f(\mathbf{S}(x))) \vee \forall x(f(x) < n) \vdash \exists x\,(f(x) = f(\mathbf{S}(x)))}$$

(D)

$$\dfrac{\begin{array}{c}(A)\\ \exists x(n+1=f(x)\wedge n+1=f(\mathbf{S}(x)))\vee\forall x(f(x)<n+1)\vdash \\ \exists x(n=f(x)\wedge n=f(\mathbf{S}(x)))\vee\forall x(f(x)<n),\\ \exists x\,(f(x)=f(\mathbf{S}(x)))\end{array}\quad(D)}{\begin{array}{c}\exists x(n+1=f(x)\wedge n+1=f(\mathbf{S}(x)))\vee\forall x(f(x)<n+1)\vdash \\ \exists x\,(f(x)=f(\mathbf{S}(x)))\end{array}}\text{cut}$$

5. Schematic Characteristic Formula of the 1-SMA Schema

We will not go into great detail concerning the construction of the following schematic NNF definition of the 1-SMA schema's cut structure. Following the method outlined in the preliminaries, one gets the following inductive definition.

Definition 5.1. Let $\mathcal{C}^*(n)$ be the clause set resulting from the following schematic NNF definition Ψ:

$$\begin{array}{rcl}
Top(0) & = & Next(0)\ \wedge\ (0=f(\mathbf{0})\vee 0=f(\mathbf{S}(\mathbf{0})))\\
Top(n+1) & = & \forall x((n+1)=f(\mathbf{S}(x))\vee f(x)<(n+1))\ \wedge\\
 & & \forall x((n+1)=f(x)\vee f(x)<(n+1))\ \wedge Next(n+1)\\
Next(0) & = & (\neg f(\mathbf{0})<0)\ \wedge\ \forall x((\neg 0=f(x))\vee(\neg 0=f(\mathbf{S}(x))))\\
Next(n+1) & = & \forall x((\neg(n+1)=f(x))\vee(\neg(n+1)=f(\mathbf{S}(x))))\ \wedge\\
 & & \forall x((\neg f(x)<(n+1))\vee n=f(x)\vee f(x)<n)\ \wedge\\
 & & \forall x((\neg f(\mathbf{S}(x))<(n+1))\vee n=f(\mathbf{S}(x))\vee f(x)<n)\ \wedge Next(n)
\end{array}$$

As is the case for all schematic characteristic clause sets $\mathcal{C}^*(n)$ is always unsatisfiable. However, as we shall show, refuting $\mathcal{C}^*(n)$ is non-trivial and rather than directly refuting $\mathcal{C}^*(n)$ itself, we can instead consider the clause set resulting from the replacement of $\forall x((n+1)=f(\mathbf{S}(x))\vee f(x)<(n+1))$ and $\forall x((n+1)=f(x)\vee f(x)<(n+1))$ by sequences of term instantiations. We will refer to the resulting clause set as $\mathcal{C}(n)$. Notice that unsatisfiability of $\mathcal{C}(n)$ implies unsatisfiability of $\mathcal{C}^*(n)$. To define $\mathcal{C}(n)$ we need a few auxiliary definitions. First of all, we need to be able to translate between the individual sort and the numeric sort.

Definition 5.2. (translation between individual and numeric sort)
Let Sw be inductively defined as follows:

$$Sw(0)\equiv\mathbf{0}\qquad\qquad Sw(n+1)\equiv\mathbf{S}(Sw(n))$$

Rather than directly refuting $\mathcal{C}^*(n)$ we consider the clause set $\mathcal{C}(n)$ which is derivable from $\mathcal{C}^*(n)$ by replacing the two clauses of $Top(n)$ by the following sequences of term instantiations.

Definition 5.3.

$$\begin{aligned} Seq_1(0,n) &\equiv (n = f(\mathbf{S}(Sw(0))) \vee f(Sw(0)) < n) \\ Seq_1(m+1,n) &\equiv (n = f(\mathbf{S}(Sw(m+1))) \vee f(Sw(m+1)) < n) \ \wedge Seq_1(m,n) \\ Seq_2(0,n) &\equiv (n = f(Sw(0)) \vee f(Sw(0)) < n) \\ Seq_2(m+1,n) &\equiv (n = f(Sw(m+1)) \vee f(Sw(m+1)) < n) \ \wedge \ Seq_2(m,n) \end{aligned}$$

Seq_1 and Seq_2 are used to define the clause set $\mathcal{C}(n) \equiv Seq_1(n,n) \ \wedge \ Seq_2(n,n) \wedge Next(n)$.

The problems we have mentioned earlier concerning loop finding methods stem from the double occurrence of the free parameter within the definitions of Seq_1 and Seq_2. Like other diagonalization arguments, stepping down one of the two instances of n is not enough to recreate the refutation we provide in the following section.

Lemma 5.4. $\mathcal{C}(n)$ is derivable from $\mathcal{C}^*(n)$.

Proof:
[sketch] Note that $\mathcal{C}(n) = Seq_1(Sw(n),n) \ \wedge \ Seq_2(Sw(n),n) \wedge Next(n)$ can be constructed from $\mathcal{C}^*(n)$ using the clauses

$$\forall x((\neg f(x) < (n+1)) \vee n = f(x) \vee f(x) < n)$$

$$\forall x((\neg f(\mathbf{S}(x)) < (n+1)) \vee n = f(\mathbf{S}(x)) \vee f(x) < n)$$

□

We perform our analysis on $\mathcal{C}(n)$ rather than $\mathcal{C}^*(n)$ because $\mathcal{C}(n)$ encodes a necessary and sufficient condition for refutability. In particular, Theorem 6.1 shows that the instantiations of $\mathcal{C}(n)$ which we have chosen are sufficient for refuting $\mathcal{C}^*(n)$, while Theorem 6.3 together with the fact that we are forced to use the atom $f(\mathbf{0}) < 0$ in any refutation provides a necessary condition for refuting the clause set.

6. The refutation of $C(n)$

In this section we provide a refutation of $C(n)$ and show that the instantiations of $C^*(n)$ are necessary and sufficient. This implies that any automated theorem prover wishing to refute this $C(n)$ is forced to follow the precise argument we outline here.

Theorem 6.1. For all $n > 1$, $\mathcal{C}(n) \vdash$ is provable in LK without cut from initial sequents of the form $A \vdash A$.

Proof:
By induction on n. For the remainder of this section *branch* refers to an application $\vee : l$. If

a branch contains both a negative and a positive instance of a literal (that is P and $\neg P$ then it may be *closed*. let $n = 1$, then

$$\mathcal{C}(1) \equiv Seq_1(\mathbf{S}(\mathbf{0}), 1) \ \wedge \ Seq_2(\mathbf{S}(\mathbf{0}), 1) \wedge Next(1)$$

and we must show that $\mathcal{C}(1) \vdash$ is provable. Unrolled, $\mathcal{C}(1)$ results in the sequent $\Gamma \vdash$ which contains the following formulas:

$$(1 = f(\mathbf{S}(\mathbf{S}(\mathbf{0}))) \vee f(\mathbf{S}(\mathbf{0})) < 1) \tag{7}$$
$$(1 = f(\mathbf{S}(\mathbf{0})) \vee f(\mathbf{0}) < 1) \tag{8}$$
$$(1 = f(\mathbf{S}(\mathbf{0})) \vee f(\mathbf{S}(\mathbf{0})) < 1) \tag{9}$$
$$(1 = f(\mathbf{0}) \vee f(\mathbf{0}) < 1) \tag{10}$$
$$\forall x((\neg 1 = f(x)) \vee (\neg 1 = f(\mathbf{S}(x)))) \tag{11}$$
$$\forall x((\neg f(x) < 1) \vee 0 = f(x) \vee f(x) < 0) \tag{12}$$
$$\forall x((\neg f(\mathbf{S}(x)) < 1) \vee 0 = f(\mathbf{S}(x)) \vee f(x) < 0) \tag{13}$$
$$(\neg f(\mathbf{0}) < 0) \tag{14}$$
$$\forall x((\neg 0 = f(x)) \vee (\neg 0 = f(\mathbf{S}(x)))) \tag{15}$$

Now let us consider the 16 sequents constructed from the clauses (7)−(10) where Δ contains clauses (11)−(15). These sequents are the leaves of the tree found in Figure 2.

$$1 = f(\mathbf{S}(\mathbf{S}(\mathbf{0}))) \ , \ 1 = f(\mathbf{S}(\mathbf{0})) \ , \ 1 = f(\mathbf{S}(\mathbf{0}))) \ , \ 1 = f(\mathbf{0}), \Delta \vdash \tag{16}$$
$$1 = f(\mathbf{S}(\mathbf{S}(\mathbf{0}))) \ , \ f(\mathbf{0}) < 1 \ , \ 1 = f(\mathbf{S}(\mathbf{0}))) \ , \ 1 = f(\mathbf{0}), \Delta \vdash \tag{17}$$
$$f(\mathbf{S}(\mathbf{0})) < 1) \ , \ 1 = f(\mathbf{S}(\mathbf{0})) \ , \ 1 = f(\mathbf{S}(\mathbf{0}))) \ , \ 1 = f(\mathbf{0}), \Delta \vdash \tag{18}$$
$$f(\mathbf{S}(\mathbf{0})) < 1) \ , \ f(\mathbf{0}) < 1 \ , \ 1 = f(\mathbf{S}(\mathbf{0}))) \ , \ 1 = f(\mathbf{0}), \Delta \vdash \tag{19}$$
$$1 = f(\mathbf{S}(\mathbf{S}(\mathbf{0}))) \ , \ 1 = f(\mathbf{S}(\mathbf{0})) \ , \ f(\mathbf{S}(\mathbf{0}))) < 1 \ , \ 1 = f(\mathbf{0}), \Delta \vdash \tag{20}$$
$$1 = f(\mathbf{S}(\mathbf{S}(\mathbf{0}))) \ , \ f(\mathbf{0}) < 1 \ , \ f(\mathbf{S}(\mathbf{0}))) < 1 \ , \ 1 = f(\mathbf{0}), \Delta \vdash \tag{21}$$
$$f(\mathbf{S}(\mathbf{0})) < 1) \ , \ 1 = f(\mathbf{S}(\mathbf{0})) \ , \ f(\mathbf{S}(\mathbf{0}))) < 1 \ , \ 1 = f(\mathbf{0}), \Delta \vdash \tag{22}$$
$$f(\mathbf{S}(\mathbf{0})) < 1) \ , \ f(\mathbf{0}) < 1 \ , \ f(\mathbf{S}(\mathbf{0}))) < 1 \ , \ 1 = f(\mathbf{0}), \Delta \vdash \tag{23}$$
$$1 = f(\mathbf{S}(\mathbf{S}(\mathbf{0}))) \ , \ 1 = f(\mathbf{S}(\mathbf{0})) \ , \ 1 = f(\mathbf{S}(\mathbf{0}))) \ , \ f(\mathbf{0}) < 1, \Delta \vdash \tag{24}$$
$$1 = f(\mathbf{S}(\mathbf{S}(\mathbf{0}))) \ , \ f(\mathbf{0}) < 1 \ , \ 1 = f(\mathbf{S}(\mathbf{0}))) \ , \ f(\mathbf{0}) < 1, \Delta \vdash \tag{25}$$
$$f(\mathbf{S}(\mathbf{0})) < 1) \ , \ 1 = f(\mathbf{S}(\mathbf{0})) \ , \ 1 = f(\mathbf{S}(\mathbf{0}))) \ , \ f(\mathbf{0}) < 1, \Delta \vdash \tag{26}$$
$$f(\mathbf{S}(\mathbf{0})) < 1) \ , \ f(\mathbf{0}) < 1 \ , \ 1 = f(\mathbf{S}(\mathbf{0}))) \ , \ f(\mathbf{0}) < 1, \Delta \vdash \tag{27}$$
$$1 = f(\mathbf{S}(\mathbf{S}(\mathbf{0}))) \ , \ 1 = f(\mathbf{S}(\mathbf{0})) \ , \ f(\mathbf{S}(\mathbf{0}))) < 1 \ , \ f(\mathbf{0}) < 1, \Delta \vdash \tag{28}$$
$$1 = f(\mathbf{S}(\mathbf{S}(\mathbf{0}))) \ , \ f(\mathbf{0}) < 1 \ , \ f(\mathbf{S}(\mathbf{0}))) < 1 \ , \ f(\mathbf{0}) < 1, \Delta \vdash \tag{29}$$
$$f(\mathbf{S}(\mathbf{0})) < 1) \ , \ 1 = f(\mathbf{S}(\mathbf{0})) \ , \ f(\mathbf{S}(\mathbf{0}))) < 1 \ , \ f(\mathbf{0}) < 1, \Delta \vdash \tag{30}$$
$$f(\mathbf{S}(\mathbf{0})) < 1) \ , \ f(\mathbf{0}) < 1 \ , \ f(\mathbf{S}(\mathbf{0}))) < 1 \ , \ f(\mathbf{0}) < 1, \Delta \vdash \tag{31}$$

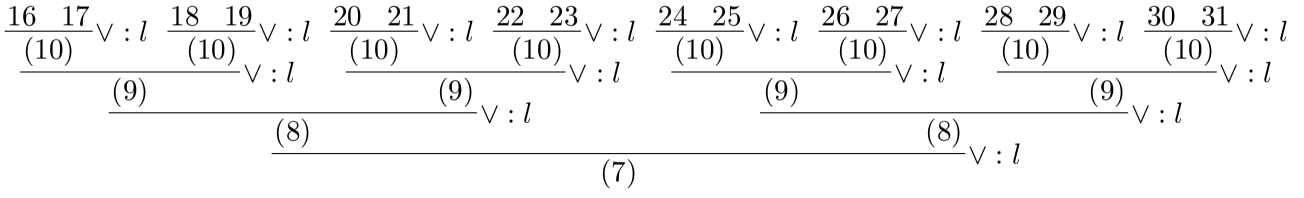

Figure 2. Proof tree construction.

If a branch contains both $1 = f(\mathbf{S}(\mathbf{0}))$ and $1 = f(\mathbf{0})$ or $1 = f(\mathbf{S}(\mathbf{0}))$ and $1 = f(\mathbf{S}(\mathbf{S}(\mathbf{0})))$ then it can be closed using (11) from Δ. For example consider branch (16)

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{1 = f(\mathbf{S}(\mathbf{0}))) \vdash 1 = f(\mathbf{S}(\mathbf{0}))}{1 = f(\mathbf{S}(\mathbf{0}))), \neg 1 = f(\mathbf{S}(\mathbf{0})), \Delta \vdash} \quad \dfrac{1 = f(\mathbf{0}) \vdash 1 = f(\mathbf{0})}{1 = f(\mathbf{0}), \neg 1 = f(\mathbf{0})), \Delta \vdash}}{1 = f(\mathbf{S}(\mathbf{0}))) \ , \ 1 = f(\mathbf{0}), (\neg 1 = f(\mathbf{0})) \vee (\neg 1 = f(\mathbf{S}(\mathbf{0}))), \Delta \vdash}}{1 = f(\mathbf{S}(\mathbf{0}))) \ , \ 1 = f(\mathbf{0}), \forall x((\neg 1 = f(x)) \vee (\neg 1 = f(\mathbf{S}(x)))), \Delta \vdash}}{1 = f(\mathbf{S}(\mathbf{S}(\mathbf{0}))) \ , \ 1 = f(\mathbf{S}(\mathbf{0})) \ , \ 1 = f(\mathbf{S}(\mathbf{0}))) \ , \ 1 = f(\mathbf{0}), \Delta \vdash}$$

This leaves the following branches:

$$1 = f(\mathbf{S}(\mathbf{S}(\mathbf{0}))) \ , \ f(\mathbf{0}) < 1 \ , \ f(\mathbf{S}(\mathbf{0}))) < 1 \ , \ 1 = f(\mathbf{0}), \Delta \vdash \tag{32}$$

$$f(\mathbf{S}(\mathbf{0})) < 1) \ , \ f(\mathbf{0}) < 1 \ , \ f(\mathbf{S}(\mathbf{0}))) < 1 \ , \ 1 = f(\mathbf{0}), \Delta \vdash \tag{33}$$

$$f(\mathbf{S}(\mathbf{0})) < 1) \ , \ 1 = f(\mathbf{S}(\mathbf{0})) \ , \ 1 = f(\mathbf{S}(\mathbf{0}))) \ , \ f(\mathbf{0}) < 1, \Delta \vdash \tag{34}$$

$$f(\mathbf{S}(\mathbf{0})) < 1) \ , \ f(\mathbf{0}) < 1 \ , \ 1 = f(\mathbf{S}(\mathbf{0}))) \ , \ f(\mathbf{0}) < 1, \Delta \vdash \tag{35}$$

$$f(\mathbf{S}(\mathbf{0})) < 1) \ , \ 1 = f(\mathbf{S}(\mathbf{0})) \ , \ f(\mathbf{S}(\mathbf{0}))) < 1 \ , \ f(\mathbf{0}) < 1, \Delta \vdash \tag{36}$$

$$f(\mathbf{S}(\mathbf{0})) < 1) \ , \ f(\mathbf{0}) < 1 \ , \ f(\mathbf{S}(\mathbf{0}))) < 1 \ , \ f(\mathbf{0}) < 1, \Delta \vdash \tag{37}$$

We can simplify these sequents by application of weakening and contraction rules so that they are all of the form

$$f(\mathbf{0}) < 1 \ , \ f(\mathbf{S}(\mathbf{0}))) < 1, \Delta \vdash \tag{38}$$

Using these two literals and (12) & (13) we derive a sequent containing the following three clauses:

$$0 = f(\mathbf{0}) \vee f(\mathbf{0}) < 0 \tag{39}$$

$$0 = f(\mathbf{S}(\mathbf{0})) \vee f(\mathbf{S}(\mathbf{0})) < 0 \tag{40}$$

$$0 = f(\mathbf{S}(\mathbf{0})) \vee f(\mathbf{0}) < 0 \tag{41}$$

We will refer to the sequent containing (39), (40), and (41) as $\mathcal{S}$ and can construct the following proof tree starting at $\mathcal{S}$

$$\dfrac{\dfrac{\dfrac{\dfrac{42 \quad 43}{(41)} \vee : l \quad \dfrac{44 \quad 45}{(41)} \vee : l}{(40)} \vee : l \quad \dfrac{\dfrac{46 \quad 47}{(41)} \vee : l \quad \dfrac{48 \quad 49}{(41)} \vee : l}{(40)} \vee : l}{(39)} \vee : l$$

Where the leaves are the following sequents

$$0 = f(\mathbf{0}), 0 = f(\mathbf{S}(\mathbf{0})), 0 = f(\mathbf{S}(\mathbf{0})), \Delta \vdash \tag{42}$$
$$0 = f(\mathbf{0}), f(\mathbf{S}(\mathbf{0})) < 0, 0 = f(\mathbf{S}(\mathbf{0})), \Delta \vdash \tag{43}$$
$$0 = f(\mathbf{0}), 0 = f(\mathbf{S}(\mathbf{0})), f(\mathbf{0}) < 0, \Delta \vdash \tag{44}$$
$$0 = f(\mathbf{0}), f(\mathbf{S}(\mathbf{0})) < 0, f(\mathbf{0}) < 0, \Delta \vdash \tag{45}$$
$$f(\mathbf{0}) < 0, 0 = f(\mathbf{S}(\mathbf{0})), 0 = f(\mathbf{S}(\mathbf{0})), \Delta \vdash \tag{46}$$
$$f(\mathbf{0}) < 0, f(\mathbf{S}(\mathbf{0})) < 0, 0 = f(\mathbf{S}(\mathbf{0})), \Delta \vdash \tag{47}$$
$$f(\mathbf{0}) < 0, 0 = f(\mathbf{S}(\mathbf{0}))f(\mathbf{0}) < 0, \Delta \vdash \tag{48}$$
$$f(\mathbf{0}) < 0, f(\mathbf{S}(\mathbf{0})) < 0, f(\mathbf{0}) < 0, \Delta \vdash \tag{49}$$

Sequent (44)−(49) contain $f(\mathbf{0}) < 0$, and thus can be closed using (14). This leaves sequents (42) and (43) which can be closed using (15). Thus we have shown that $\mathcal{C}(1) \vdash$ is provable. For the step case we assume that $\mathcal{C}(n) \vdash$ is provable and show that $\mathcal{C}(n+1) \vdash$ is provable.

Let us consider the sequent $Seq_1(Sw(n), n), Seq_2(Sw(n), n), Next(n) \vdash$ which can be constructed from $\mathcal{C}^*(n+1) \vdash$ using the clauses

$$\forall x((\neg f(x) < (n+1)) \vee n = f(x) \vee f(x) < n) \tag{50}$$
$$\forall x((\neg f(\mathbf{S}(x)) < (n+1)) \vee n = f(\mathbf{S}(x)) \vee f(x) < n) \tag{51}$$

found in $Next(n+1)$. We denote the following literal set by Γ:

$$f(\mathbf{0}) < (n+1)\ ,\ f(\mathbf{S}(\mathbf{0}))) < (n+1)\ ,\ \cdots\ ,\ f(\mathbf{S}(Sw(n))) < (n+1) \tag{52}$$

These observations entail the soundness of the following proof tree:

$$\dfrac{\dfrac{Seq_1(Sw(n), n), Seq_2(Sw(n), n), Next(n) \vdash}{\vdots}}{\Gamma, Next(n+1) \vdash}$$

Notice the similarities between $\Gamma, Next(n+1) \vdash$ and (38). Essentially, if we consider the sequent $Seq_1(Sw(n+1), n+1), Seq_2(Sw(n+1), n+1), Next(n+1) \vdash$, apply $\vee : l$ to each clause of $Seq_1(Sw(n+1), n+1)$ and $Seq_2(Sw(n+1), n+1)$ one, close all branches where $\forall x((\neg(n+1) = f(x)) \vee (\neg(n+1) = f(\mathbf{S}(x))))$ can be applied, and finally apply weakenings and contraction as in the basecase $\Gamma, Next(n+1) \vdash$ would be the resulting sequent. To see that this would be the case we would have to show that the sequents at the leaves of this construction can either be closed by $\forall x((\neg(n+1) = f(x)) \vee (\neg(n+1) = f(\mathbf{S}(x))))$ or they contain Γ. This follows directly from the definition of Seq_1 and Seq_2. Notice that $Seq_1(k, n+1)$ and $Seq_2(k, n+1)$ for some $0 \leq k \leq n+1$ add the following two formula to the sequent:

$$(n+1) = f(\mathbf{S}(k)) \vee f(k) < (n+1) \tag{53}$$
$$(n+1) = f(k) \vee f(k) < (n+1) \tag{54}$$

From these two formulas we can make the following four sequents:

$$(n+1) = f(\mathbf{S}(k)), (n+1) = f(k), \Delta \vdash \tag{55}$$
$$(n+1) = f(\mathbf{S}(k)), f(k) < (n+1), \Delta \vdash \tag{56}$$
$$f(k) < (n+1), f(k) < (n+1), \Delta \vdash \tag{57}$$
$$f(k) < (n+1), (n+1) = f(k), \Delta \vdash \tag{58}$$

Notice that only (55) can be closed by $\forall x((\neg(n+1) = f(x)) \vee (\neg(n+1) = f(\mathbf{S}(x))))$, but every other branch contains at least one instance of $(n+1) = f(k)$. Thus, every branch which is not closed will contain at least one instance $(n+1) = f(k)$ for each $0 \leq k \leq n+1$. This is precisely Γ. Thus it follows that the proof tree

$$\dfrac{\dfrac{\dfrac{Seq_1(Sw(n), n), Seq_2(Sw(n), n), Next(n) \vdash}{\vdots}}{\Gamma, Next(n+1) \vdash}}{\vdots}$$
$$\overline{Seq_1(Sw(n+1), n+1), Seq_2(Sw(n+1), n+1), Next(n+1) \vdash}$$

is a sound derivation. The root of this derivation is precisely $\mathcal{C}(n+1) \vdash$. By the induction hypothesis that $\mathcal{C}(n) \vdash$ is provable which is the sequent found on every open branch we see that $\mathcal{C}(n+1) \vdash$ is also provable. □

Note that we are considering provability of a negative statement in LK without cut rather than provability by resolution. Essentially, these questions are the same. Furthermore, we only consider provability of instances rather than provability of the entire sequence at once. This is justified by the fact that sequence provability puts additional restrictions on the proof construction, for example finite representability. This is also what makes provability of the 1-SMA schema's characteristic formula so hard, one needs n instances of a clause indexed by n which implies that a non-finite number of inferences needs to be performed in a finite number of steps.

Theorem 6.1 shows that the instantiations of $\mathcal{C}(n)$ are sufficient for showing the unsatisfiability of $\mathcal{C}^*(n)$. To show that these instantiations are also necessary let us consider a restriction of $C(n)$ which only contains $2n-1$ of the $2n$ instances derived from $\mathcal{C}^*(n)$. This results in a sequence of clause sets we refer to as $C_1(n,k)$ and $C_2(n,k)$, where $0 \leq k \leq n$, which use the following variants of Seq_1 and Seq_2.

Definition 6.2.

$$\begin{aligned}
Seq_1'(k+1,n,k+1) &\equiv Seq_1'(k,n,k+1)\\
Seq_1'(m+1,n,k) &\equiv (n = f(\mathbf{S}(sw(m+1))) \vee f(sw(m+1)) < n) \ \wedge \ Seq_1'(m,n,k)\\
Seq_1'(0,n,0) &\equiv \top\\
Seq_1'(0,n,k) &\equiv (n = f(\mathbf{S}(sw(0))) \vee f(sw(0)) < n)\\
Seq_2'(k+1,n,k+1) &\equiv Seq_2'(k,n,k+1)\\
Seq_2'(m+1,n,k) &\equiv (n = f(sw(m+1)) \vee f(sw(m+1)) < n) \ \wedge \ Seq_2'(m,n,k)\\
Seq_2'(0,n,0) &\equiv \top\\
Seq_2'(0,n,k) &\equiv (n = f(sw(0)) \vee f(sw(0)) < n)
\end{aligned}$$

We define $C_1(n,k) = Seq_1'(k, Sw(n), n) \ \wedge \ Seq_2(Sw(n), n) \wedge Next(n)$ and $C_2(n,k) = Seq_1(Sw(n), n) \wedge Seq_2'(k, Sw(n), n) \wedge Next(n)$. Notice that for $Seq_1'(m,n,k)$ and $Seq_2'(m,n,k)$ we skip the instantiation when $m = k$.

Theorem 6.3. For all $n > 1$ and $0 \leq k \leq n$, $C_1(n,k) \vdash$ and $C_2(n,k) \vdash$ are not provable in LK without cut from initial sequents of the form $A \vdash A$.

Proof:
By induction on n and by case distinction on k. let $n = 1$, then we are considering the following four sequents $\mathcal{C}_1(1,0) \vdash$, $\mathcal{C}_2(1,0) \vdash$, $\mathcal{C}_1(1,1) \vdash$, $\mathcal{C}_2(1,1) \vdash$. Where Δ denotes clauses (11)−(15), we can represent the sequents as follows:

$$seq_1'(Sw(1),1,0) \ , \ Seq_2(Sw(1),1), \Delta \vdash \tag{59}$$
$$seq_1'(Sw(1),1,1) \ , \ Seq_2(Sw(1),1), \Delta \vdash \tag{60}$$
$$Seq_1(Sw(1),1) \ , \ Seq_2'(Sw(1),1,0), \Delta \vdash \tag{61}$$
$$Seq_1(Sw(1),1) \ , \ Seq_2'(Sw(1),1,1), \Delta \vdash \tag{62}$$

We can replace seq_1 ,seq_1', seq_2, and seq_2' by formula as follows:

(59)

$$(1 = f(\mathbf{S}(\mathbf{S}(\mathbf{0}))) \vee f(\mathbf{S}(\mathbf{0})) < 1) \tag{63}$$
$$(1 = f(\mathbf{S}(\mathbf{0})) \vee f(\mathbf{S}(\mathbf{0})) < 1) \tag{64}$$
$$(1 = f(\mathbf{0}) \vee f(\mathbf{0}) < 1) \tag{65}$$

(60)

$$(1 = f(\mathbf{S}(\mathbf{0})) \vee f(\mathbf{0}) < 1) \tag{66}$$
$$(1 = f(\mathbf{S}(\mathbf{0})) \vee f(\mathbf{S}(\mathbf{0})) < 1) \tag{67}$$
$$(1 = f(\mathbf{0}) \vee f(\mathbf{0}) < 1) \tag{68}$$

(61)

$$(1 = f(\mathbf{S}(\mathbf{S}(\mathbf{0}))) \vee f(\mathbf{S}(\mathbf{0})) < 1) \tag{69}$$
$$(1 = f(\mathbf{S}(\mathbf{0})) \vee f(\mathbf{0}) < 1) \tag{70}$$
$$(1 = f(\mathbf{S}(\mathbf{0})) \vee f(\mathbf{S}(\mathbf{0})) < 1) \tag{71}$$

(62)

$$(1 = f(\mathbf{S}(\mathbf{S}(\mathbf{0}))) \vee f(\mathbf{S}(\mathbf{0})) < 1) \tag{72}$$
$$(1 = f(\mathbf{S}(\mathbf{0})) \vee f(\mathbf{0}) < 1) \tag{73}$$
$$(1 = f(\mathbf{0}) \vee f(\mathbf{0}) < 1) \tag{74}$$

As was done in Theorem 6.1 we can apply $\vee : l$ to each of the formula derived above and consider which branches can be closed by formula of Δ and which cannot be. In this case Δ is $next(1)$. Below we highlight the branches derived from the above sequents which cannot be closed using (11).

(59)

$$f(\mathbf{S}(\mathbf{0})) < 1 \ , \ f(\mathbf{S}(\mathbf{0})) < 1 \ , \ 1 = f(\mathbf{0}), \Delta \vdash \tag{75}$$

(60)

$$f(\mathbf{0}) < 1 \ , \ 1 = f(\mathbf{S}(\mathbf{0})) \ , \ f(\mathbf{0}) < 1, \Delta \vdash \tag{76}$$

(61)

$$f(\mathbf{S}(\mathbf{0})) < 1 \ , \ 1 = f(\mathbf{S}(\mathbf{0})) \ , \ f(\mathbf{S}(\mathbf{0})) < 1, \Delta \vdash \tag{77}$$

(62)

$$1 = f(\mathbf{S}(\mathbf{S}(\mathbf{0}))) \ , \ f(\mathbf{0}) < 1 \ , \ f(\mathbf{0}) < 1, \Delta \vdash \tag{78}$$

unlike the basecase in of Theorem 1 applying weakening and contraction results in more than one branch. In particular, this results in the following:

(60) & (62)

$$f(\mathbf{0}) < 1, \Delta \vdash \tag{79}$$

(59) & (61)

$$f(\mathbf{S}(\mathbf{0})) < 1, \Delta \vdash \tag{80}$$

Now let us consider the result of apply (12) and (13) to (79) and (80). The result is as follows

(60) & (62)

$$0 = f(\mathbf{0}) \vee f(\mathbf{0}) < 0, \Delta \vdash \tag{81}$$

(59) & (61)

$$(0 = f(\mathbf{S}(\mathbf{0})) \vee f(\mathbf{0}) < 0), (0 = f(\mathbf{S}(\mathbf{0})) \vee f(\mathbf{S}(\mathbf{0})) < 0), \Delta \vdash \tag{82}$$

The branch $0 = f(\mathbf{0}), \Delta \vdash$ entails from (81) and cannot be closed. Also the branch $0 = f(\mathbf{S}(\mathbf{0})), f(\mathbf{S}(\mathbf{0})) < 0, \Delta \vdash$ and cannot be closed. Thus, none of the sequents $\mathcal{C}_1(1,0) \vdash$, $\mathcal{C}_2(1,0) \vdash$, $\mathcal{C}_1(1,1) \vdash$, $\mathcal{C}_2(1,1) \vdash$ are provable.

Now let use assume that the sequents $\mathcal{C}_1(n,k) \vdash$ and $\mathcal{C}_2(n,k) \vdash$ are not provable for each $0 \leq k \leq n$, we now show that the clause sets $\mathcal{C}_1(n+1,k) \vdash$ and $\mathcal{C}_2(n+1,k) \vdash$ are not provable for any $0 \leq k \leq n+1$.

We prove the step case by a case distinction on k. when $k = n+1$, the missing clause from $\mathcal{C}_1(n+1, n+1)$ is

$$n+1 = f(\mathbf{S}(sw(n+1))) \vee f(sw(n+1)) < n+1 \tag{83}$$

and the missing clause of $\mathcal{C}_2(n+1, n+1)$ is

$$n+1 = f(sw(n+1)) \vee f(sw(n+1)) < n+1 \tag{84}$$

In the case of $\mathcal{C}_1(n+1, n+1)$ the only clause containing $f(sw(n+1)) < n+1$ is (84) and In the case of $\mathcal{C}_2(n+1, n+1)$ the only clause containing $f(sw(n+1)) < n+1$ is (83). Now consider the branches which have the following form:

$$f(\mathbf{0}) < n+1\ ,\ \cdots\ ,\ f(sw(n)) < n+1\ ,\ n+1 = f(sw(n+1)) \tag{85}$$

$$f(\mathbf{0}) < n+1\ ,\ \cdots\ ,\ f(sw(n)) < n+1\ ,\ n+1 = f(\mathbf{S}(sw(n+1))) \tag{86}$$

Being that neither branch contains the literal $f(sw(n+1)) < n+1$ we cannot derive the following formula:

$$n = f(sw(n+1)) \vee f(sw(n+1)) < n \tag{87}$$

$$n = f(\mathbf{S}(sw(n))) \vee f(sw(n)) < n \tag{88}$$

While (87) isn't really of interest $n = f(\mathbf{S}(sw(n))) \vee f(sw(n)) < n$ is a clause of $\mathcal{C}(n)$ which cannot be derived from

$$f(\mathbf{0}) < n+1 \ , \ \cdots \ , \ f(sw(n)) < n+1$$

and thus on this branch we can only derive $\mathcal{C}_1(n,n) \vdash$ and by the induction hypothesis $\mathcal{C}_1(n,n) \vdash$ is not provable. Now what is left to show is that the same can be done of $k < n+1$. The missing clause from $\mathcal{C}_1(n+1,k)$ is

$$n+1 = f(\mathbf{S}(sw(k))) \vee f(sw(k)) < n+1 \tag{89}$$

and the missing clause of $\mathcal{C}_2(n+1,k)$ is

$$n+1 = f(sw(k)) \vee f(sw(k)) < n+1 \tag{90}$$

Like in the previous case, for $\mathcal{C}_1(n+1,k)$ the only clause containing $f(sw(k)) < n+1$ is (90) and in the case of $\mathcal{C}_2(n+1,k)$ the only clause containing $f(sw(k)) < n+1$ is (89). Now consider the branches which have the following form:

$$f(\mathbf{0}) < n+1 \ , \ \cdots \ , \ n+1 = f(sw(k)) \ , \ \cdots \ , \ f(sw(n+1)) < n+1 \tag{91}$$
$$f(\mathbf{0}) < n+1 \ , \ \cdots \ , \ n+1 = f(\mathbf{S}(sw(k))) \ , \ \cdots \ , \ f(sw(n+1)) < n+1 \tag{92}$$

Being that neither branch contains the literal $f(sw(k)) < n+1$ we cannot derive the following clauses:

$$n = f(sw(k)) \vee f(sw(k)) < n \tag{93}$$
$$n = f(\mathbf{S}(sw(k-1))) \vee f(sw(k-1)) < n \tag{94}$$

If $k = 0$ we only need to consider line (93) in which case we can only derive $\mathcal{C}_2(n,0)$ on the branch which is satisfiable by the induction hypothesis. If $0 < k < n+1$ then we can only derive the sequent

$$Seq'_1(Sw(n),n,k) \ \wedge \ Seq'_2(Sw(n),n,k) \wedge Next(n) \vdash$$

which is missing two clauses instead of just one and being that both $\mathcal{C}_1(n,k)$ and $\mathcal{C}_2(n,k)$ are not provable a subset of either one of them is also not provable. Thus we have proven the theorem by induction. □

This result not only provides a necessary and sufficient set of instantiations for refutability, it also provides the minimal refutation of the characteristic formula defined above. This minimal refutation also happens to be unique modulo renaming.

7. Conclusion: The Issue with Loop Discovery Methods *á la* Aravantinos *et al.*

Loop discovery mechanisms fail because proving the unsatisfiability of $C(n)$ requires discovering the hidden loop within the quantifier instantiations and then discovering a second loop over this hidden one. In essence, this amounts to diagonalization-like argument separating loop_1 from loop_2. Note that the structure of the internal loop is dependent on the external one and thus illustrates the precise relationship between the two occurrences of n in Seq_1 and Seq_2. This translates to an inability to step down without instantiating the quantifier with a term which makes reference to the free parameter independently of other occurrences of the free parameter. This supports observations that such methods are inherently Σ_1-incomplete [24] as well as weakness of methods such as [20] concerning the analysis of mathematical proofs. Note that this does not entirely rule out loop discovery methods for automated theorem proving of recursive clause sets but it does highlight limitations of current techniques. Unfortunately, earlier work by the same authors provide hard upper limits for the propositional version of the method which will in all likelihood carry over the the first-order version [21].

This leaves an interesting question, that is whether or not this example represents a hard upper limit for loop discovery methods for inductive theorem proving. The same clause set discussed here was given to Viper [31] a tree grammar based inductive prover [10] which after much work ($\sim$ 5 hours) was able to find the following invariant.

$$(Top(n) \to 0 = f(\mathbf{S}(\mathbf{0}))) \ \wedge \ (Top(n) \ \to \ 0 = f(\mathbf{0})) \wedge$$
$$(Top(n) \to Next(0)) \ \wedge \ \neg(Next(n+1) \ \wedge \ Next(n) \ \wedge \ Top(n+1))$$

An interesting question left unanswered is how much harder can recursive NNF formula indexed by a single free parameter over a monadic signature get, in particular if we restrict them to the cut structure of a single parameter schematic proof which limits the relations between free variables. We plan to investigate this in future work. Note that it is unlikely that k-SMA will be much harder than 1-SMA, thus, a more complex recursive NNF formula will most likely need to be considered, possibly a different variant of the infinitary pigeonhole principle.

References

[1] Dunchev C, Leitsch A, Rukhaia M, Weller D. Cut-Elimination and Proof Schemata. In: Logic, Language, and Computation; 2013. p. 117–136.

[2] Miller DA. A compact representation of proofs. Studia Logica. 1987;46(4):347–370.

[3] Brotherston J. Cyclic Proofs for First-Order Logic with Inductive Definitions. In: Tableaux'05. vol. 3702 of Lecture Notes in Comp. Sci.; 2005. p. 78–92.

[4] Brotherston J, Simpson A. Sequent calculi for induction and infinite descent. Journal of Logic and Computation. 2010;21(6):1177–1216.

[5] Cerna DM, Lettmann M. Integrating a Global Induction Mechanism into a Sequent Calculus. In: Schmidt RA, Nalon C, editors. Automated Reasoning with Analytic Tableaux and Related Methods. Cham: Springer International Publishing; 2017. p. 278–294.

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

[7] Baaz M. Note on the generalization of calculations. Theoretical Computer Science. 1999;224(1–2):3 – 11. Available from: http://www.sciencedirect.com/science/article/pii/S0304397598003041.

[8] Althaus E, Kruglov E, Weidenbach C. Superposition Modulo Linear Arithmetic SUP(LA). In: Ghilardi S, Sebastiani R, editors. Frontiers of Combining Systems. Berlin, Heidelberg: Springer Berlin Heidelberg; 2009. p. 84–99.

[9] Comon H. Inductionless Induction. In: Robinson JA, Voronkov A, editors. Handbook of Automated Reasoning (in 2 volumes). Elsevier and MIT Press; 2001. p. 913–962.

[10] Eberhard S, Hetzl S. Inductive theorem proving based on tree grammars. Ann Pure Appl Logic. 2015;166(6):665–700. Available from: https://doi.org/10.1016/j.apal.2015.01.002.

[11] Giesl J, Kapur D. Decidable Classes of Inductive Theorems. In: Proceedings of the First International Joint Conference on Automated Reasoning. IJCAR '01. London, UK, UK: Springer-Verlag; 2001. p. 469–484. Available from: http://dl.acm.org/citation.cfm?id=648237.753948.

[12] Hetzl S, Wong TL. Restricted notions of provability by induction. 2017;abs/1704.01930. Available from: http://arxiv.org/abs/1704.01930.

[13] Horbach M, Weidenbach C. Deciding the Inductive Validity of $\forall\exists^*$ Queries. In: Grädel E, Kahle R, editors. Computer Science Logic. Berlin, Heidelberg: Springer Berlin Heidelberg; 2009. p. 332–347.

[14] Kapur D, Subramaniam M. Extending Decision Procedures with Induction Schemes. In: McAllester D, editor. Automated Deduction - CADE-17. vol. 1831 of Lecture Notes in Computer Science. Springer Berlin Heidelberg; 2000. p. 324–345. Available from: http://dx.doi.org/10.1007/10721959_26.

[15] Peltier N. A paramodulation-based calculus for refuting schemata of clause sets defined by rewrite rules. Journal of Logic and Computation. 2017;27(2):549–576. Available from: http://dx.doi.org/10.1093/logcom/exu078.

[16] Weidenbach C. In: Meyer R, Platzer A, Wehrheim H, editors. Automated Reasoning Building Blocks. Cham: Springer International Publishing; 2015. p. 172–188. Available from: https://doi.org/10.1007/978-3-319-23506-6_12.

[17] Cerna DM, Leitsch A. 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; 2016. p. 241–256.

[18] Kovács L, Voronkov A. First-Order Theorem Proving and Vampire. In: Sharygina N, Veith H, editors. Computer Aided Verification. vol. 8044 of Lecture Notes in Computer Science. Springer Berlin Heidelberg; 2013. p. 1–35.

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

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

[21] Aravantinos V, Caferra R, Peltier N. Decidability and undecidability results for propositional schemata. Journal of Artificial Intelligence Research. 2011 Jan;40(1):599–656. Available from: http://dl.acm.org/citation.cfm?id=2016945.2016961.

[22] Cerna D. A Tableaux-Based Decision Procedure for Multi-parameter Propositional Schemata. In: Intelligent Computer Mathematics. vol. 8543 of Lecture Notes in Comp. Sci. Springer; 2014. p. 61–75.

[23] Aravantinos V, Echenim M, Peltier N. A Resolution Calculus for First-order Schemata. Fundam Inf. 2013 Apr;125(2):101–133. Available from: http://dx.doi.org/10.3233/FI-2013-855.

[24] Vierling J. Cyclic Superposition and Induction. Technical University of Vienna; 2018.

[25] Sutcliffe G. The TPTP Problem Library and Associated Infrastructure - From CNF to TH0, TPTP v6.4.0. J Autom Reasoning. 2017;59(4):483–502. Available from: https://doi.org/10.1007/s10817-017-9407-7.

[26] Claessen K, Johansson M, Rosén D, Smallbone N. TIP: Tons of Inductive Problems. In: Intelligent Computer Mathematics - International Conference, CICM 2015, Washington, DC, USA, July 13-17, 2015, Proceedings; 2015. p. 333–337. Available from: https://doi.org/10.1007/978-3-319-20615-8_23.

[27] Mendelson E. Introduction to Mathematical Logic. 5th ed. Chapman & Hall/CRC; 2009.

[28] Takeuti G. Proof Theory. vol. 81 of Studies in logic and the foundations of mathematics. American Elsevier Pub.; 1975.

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

[30] Cerna DM, Lolic A. Proof Schemata for Theories equivalent to PA: on the Benefit of Conservative Reflection Principles. RISC; 2018. In review. Available from: https://arxiv.org/abs/1711.10994.

[31] Eberhard S, Ebner G, Hetzl S. Algorithmic Compression of Finite Tree Languages by Rigid Acyclic Grammars. ACM Trans Comput Log. 2017;18(4):26:1–26:20. Available from: http://doi.acm.org/10.1145/3127401.