

On the Complexity of Unsatisfiable Primitive Recursively defined Σ_1 -Sentences

David M. Cerna¹²

¹ Research Institute for Symbolic Computation, Johannes Kepler University Linz,
Austria

² Institute for Formal Models and Verification, Johannes Kepler University Linz,
Austria

david.cerna@risc.jku.at

Abstract. We introduce a measure of complexity based on formula occurrence within instance proofs of an inductive statement. Our measure is closely related to *Herbrand Sequent length*, but instead of capturing the number of necessary term instantiations, it captures the finite representational difficulty of a recursive sequence of proofs. We restrict ourselves to a class of unsatisfiable primitive recursively defined negation normal form first-order sentences, referred to as *abstract sentences*, which capture many problems of interest; for example, variants of the *infinitary pigeonhole principle*. This class of sentences has been particularly useful for inductive formal proof analysis and proof transformation. Together our complexity measure and abstract sentences allow use to capture a notion of *tractability* for state-of-the-art approaches to inductive theorem proving, in particular *loop discovery* and *tree grammar* based inductive theorem provers. We provide a complexity analysis of an important abstract sentence, and discuss the analysis of a few related sentences, based on the infinitary pigeonhole principle which we conjecture represent the upper limits of tractability and foundation of intractability with respect to the current approaches.

1 Introduction

Defining what it means for a logical sentence to be “difficult to prove” provides a baseline from which relative difficulty between problems may be measured. Furthermore, such a measure enables one to study the performance of a given theorem proving method with respect to a particular type of logical sentence. A quite simple example of such a measure of difficulty is provided by the TPTP library [34]. Each problem file contains a header including the percentage of theorem provers (included in system on TPTP) which can solve the problem. Without taking theorem provers themselves into account one can always use as a measure of difficulty the size of the smallest proof of the statement in a particular formalism (This form of complexity is a fundamental notion within the field of proof complexity [7, 17]). A closely related notion concerns the difficulty of proving a skolemized first-order sentence. Its complexity can be measured by considering the length of its shortest *mid-sequent* [35], a corollary of Herbrand’s

theorem and cut-elimination. Propositionally, one could consider a sentence’s structural distance for a maximally complex sentence such as the propositional *Pigeonhole Principle*.

Notice how each of these measures is in some way related to the size of a statement’s proof. While this works as a measure of complexity concerning the difficulty of proof search within most propositional and first-order theories, this measure fails to capture essential aspects of theories which include some form of induction, namely that the “shape” of the proof also plays a role. What we are referring to by shape is that part of the proof must take the role of the induction invariant and in doing so must provide a uniform construction parametrized by the variable over which induction is being performed. Often this “shape” can be formally presented combinatorially. This hints at an application of this work to combinatorial enumerations which we plan to investigate in the future. Rather, then simply referring to these notions as the induction invariant, we take this more abstract description in order to emphasize what we are trying to capture. We are particularly interested in complexity measures which will benefit provers based on instance analysis and thus focus on notions which describe invariants within such structures.

In our work the notion of shape captures the complexity of constructing the necessary term tuples needed for constructing a resolution refutation, and thus our complexity measure borrows concepts from the theory of recursive functions. More specifically, we investigate a recursive function hierarchy defined by asymptotic computational complexity [22] and use it to provide a measure of the difficulty of refuting unsatisfiable primitive recursively defined Σ_1 -sentences.

2 Background and Related Work

Our choice to investigate primitive recursively defined Σ_1 -sentences is motivated by *schematic proof analysis* [15, 25] using a variant of the CERES method of cut elimination [4, 5]. This form of proof analysis transforms the cut structure of a formal proof into a *negation normal form (NNF)* formula which is unsatisfiable. Discovering a proof of the cut structure’s unsatisfiability allows one to reconstruct the initial proof so that it only contains atomic cuts. The schematic version of the method extracts a primitive recursive formula, representing the cut-structure, from a primitive recursively defined proof. This formula is unsatisfiable for every evaluation of its primitive recursive definition, however, formally proving this fact requires an inductive argument.

The current approach (used in [5] and formalized in [15]) to finding a proof showing that the primitive recursive formula representation of the cut structure is indeed unsatisfiable is to perform an instance analysis and attempt to produce a “proof construction” invariant. In [25], the inductive superposition prover of V. Aravantinos *et al.* [3] was used to provide this “proof construction” invariant.

When we say provide a proof construction method, what we actually mean is provide a recursive function with enumerative properties with respect to the term structure. When we phrase the problem in this manner the limits of tractability

and the concept of difficulty become much clearer. However, what remains elusive is how to connect this to a given primitive recursive formula. This is the issue we want to elucidate with this work.

Concerning, applicability of our complexity measure to existing inductive theorem provers, at first the scope might seem quite limited given that our developments focused on three particularly special approaches to the problem, namely cyclic reasoning [10], superposition resolution [3], and instance analysis via tree grammars [19]. This specificity fails to take into account the concept of difficulty for some of the most prominent inductive provers such as HipSpec [16], ACL2s [8], and Zeno [33], however, we argue that our approach to the shape problem is fundamental enough that the concept provides an interpretation of difficult which covers most approaches to the problem. Such prominent inductive provers, as mentioned above, will need to find the term constructions we discuss and will still needed to describe the “shape” construction, albeit differently.

3 A Primitive Recursive Formula Construction

The complexity theory we develop in this paper considers a particular type of primitive recursive formula construction based on an indexing sort (in our case numerals) and an individual sort. This type of formula has been studied in previous work [2, 3, 12, 25], and is closely related to proof transformation in the presence of induction and cyclic proof theory [9, 11]. Furthermore, V. Aravantinos *et al.* [3] introduced a powerful superposition based prover for formula definitions closely related to the construction we introduce in this work. However, little may be found in literature concerning this construction, and thus, the particular theory handled by their prover is unknown. Recent work points to the Σ_1 -incompleteness of their prover [36], and thus motivates the categorization of unsatisfiable Σ_1 -sentences discussed in this work.

We consider a two sorted first-order language \mathcal{L} . The sorts are ω and ι , that is the *numeric* and *individual* sort, respectively. The numeric sort contains all terms constructed from the signature $\{0, s(\cdot)\}$ and a countably infinite set of variables \mathcal{V}_ω . Ground terms of the ω sort will be referred to as numerals and are ordered using the total linear ordering of the natural numbers denoted by $<$. We denote the set of all ω terms by Ω and the set of all ground ω terms by Ω_G . The numeric sort will be used to index term and formula occurrences within primitive recursive definitions. While restricting ourselves to such a simple term language is not necessary, the additional complications of complex inductive structures, such as list, will not aid our exposition.

Terms of the ι sort are constructed using a finite signature Σ_ι of fixed arity function symbols and a countably infinite set of variables \mathcal{V}_ι . Essentially ι represents the standard term language of first-order predicate logic. We will abbreviate sequences of terms of the ι sort t_1, \dots, t_n by \bar{t} . Similar will be done for the ω but instead of lowercase Latin characters we use lowercase Greek characters, i.e. $\alpha_1, \dots, \alpha_n$ will be denoted by $\bar{\alpha}$.

Formula are constructed from a countably infinite set of predicate symbols \mathcal{P} with fixed arity. Note that predicates may take arguments of both the ω and ι sort and thus, as a rule ι sort arguments will precede the arguments of the ω sort. In addition to predicate symbols, formula may also contain *context variables* from a countably infinite set \mathcal{V}_{ctx} denoted by \bullet, \star, \dots .

In order to differentiate between the various free variable types occurring within a given formula or term, we define $\mathcal{V}_x(F)$ where $x \in \{\omega, \iota, ctx\}$ which denotes the set of free variables of the given type within F . We will refer to a formula F as *ι -closed* if $\mathcal{V}_\iota(F) = \emptyset$, *context-free* if $\mathcal{V}_{ctx}(F) = \emptyset$, and *static* if $\mathcal{V}_\omega(F) = \emptyset$. The following construction defines F_0 , the set of *0-abstract formula*:

Definition 1 (0-Abstract Formula). *The set F_0 contains formula constructed using the following inductive definition:*

- $\star \in \mathcal{V}_{ctx}$, then $(\star, \emptyset) \in F_0$.
- $P \in \mathcal{P}$ with arity n , and $\bar{\alpha}, \bar{t}$ are tuples of terms of appropriate length and type, then $(P(\bar{t}, \bar{\alpha}), \emptyset), (\neg P(\bar{t}, \bar{\alpha}), \emptyset) \in F_0$.
- $(P, \emptyset), (Q, \emptyset) \in F_0$ and $\mathcal{V}_{ctx}(Q) \cap \mathcal{V}_{ctx}(P) = \emptyset$, then $(P \wedge Q, \emptyset), (P \vee Q, \emptyset) \in F_0$.
- $(F(x), \emptyset) \in F_0$ and $x \in \mathcal{V}_\iota(F(x))$ then $(\forall x F(x), \emptyset) \in F_0$. We refer to x as being bound by the quantifier $\forall x$.

Once a quantifier binds a variable the variable will be considered *bound* rather than *free*. The set of bound variables of a formula $F \in F_0$ will be denoted by $bd(F)$ and $bd(F) \cap \mathcal{V}_\iota(F) \equiv \emptyset$. Only variables of the ι sort may be bound.

The formula construction provided in Definition 1 serves as the foundation for our hierarchy of primitive recursively defined formula. Unfortunately, unlike primitive recursive function definitions, if we want to capture all the desired behavior precisely, we need to develop a slightly more involved construction.

However, before we define so called *n-abstract formula* we need to distinguish a few important categories. The set of context-free *n-abstract formula* are referred to as *essential* and otherwise F is referred to as *expandable*. If the *n-abstract formula* are also ι -closed they are referred to as *sentences*. Note that for any *expandable formula* F , each variable of $\mathcal{V}_{ctx}(F)$ may occur once in F .

From F_0 together with countably infinite set of defined predicate symbols \mathcal{P} (defined predicate symbols will be denoted by a $\hat{\cdot}$) we may construct *schematic 0-abstract formula* which represent an infinite sequence of essential 0-abstract formula. Normalization of schematic 0-abstract formula is performed by replacing free occurrences of ω variables by ground terms of the ω sort producing an essential 0-abstract formula. Rather than defining schematic 0-abstract formula we provide a definition of the more general concept.

Definition 2 (Schematic n-Abstract Formula). *Let $(G, P) \in F_n$ be essential and $(H, Q) \in F_n$ expandable, where $l \in \mathcal{V}_\omega(H)$, $l \notin \mathcal{V}_\omega(G)$, $\mathcal{V}_{ctx}(H) =$*

$\{\star_1, \dots, \star_k\}$, m is a fresh ω variable, and \hat{F} is fresh defined predicate symbol. Let P' be the set of normalization rules $P' =$

$$\left\{ \begin{array}{l} \hat{F}(\bar{x}, \bar{\alpha}, s(m)) \Rightarrow H(\bar{x}, \bar{\alpha}, l) \left[\star_1 \setminus \hat{F}(\bar{s}_1, \bar{\alpha}_1, m), \dots, \star_k \setminus \hat{F}(\bar{s}_k, \bar{\alpha}_k, m) \right] \{l \leftarrow s(m)\} \\ \hat{F}(\bar{x}, \bar{\alpha}, 0) \Rightarrow G(s_{k+1}, \alpha_{k+1}) \end{array} \right\}$$

where $\bar{s}_1, \dots, \bar{s}_{k+1}$ are sequences of individual terms of appropriate length, $\bar{\alpha}_1, \dots, \bar{\alpha}_{k+1}$ are sequences of ω terms of appropriate length which does not contain l . Then $(\hat{F}(\bar{t}', \bar{\alpha}', \beta), P' \cup P \cup Q)$ is a schematic n -abstract formula, where $\beta, \bar{\alpha}' \in \Omega$ and \bar{t}' is a sequence of individual sort terms. The set of all schematic n -abstract formula will be denoted by \mathcal{S}_n .

n -abstract formula differ slightly from 0-abstract formula in that they may contain schematic k -abstract formula for $k < n$.

Definition 3 (n -Abstract Formula). The set F_n contains formula constructed using the following inductive definition:

- $(F, P) \in F_k \cup S_k$ for $0 \leq k < n$, then $(F, P) \in F_n$.
- $(F, P), (F', Q) \in F_n$ and $\mathcal{V}_{ctx}(F) \cap \mathcal{V}_{ctx}(F') = \emptyset$, then $(F \wedge F', P \cup Q), (F \vee F', P \cup Q) \in F_n$.
- $(F(x), P) \in F_n$ and $x \in \mathcal{V}_l(F(x))$ then $(\forall x F(x), P) \in F_n$ where $\forall x$ binds all free occurrences of x in $F(x)$.

Thus, the set of all abstract formula, denoted by \mathcal{L}_F^ω , is equivalent to $\bigcup_{i=0}^{\infty} F_i$. As we shall see this is quite an expressive class of formula even though one cannot define recursive function symbols directly. The formula we discuss here are quite similar to those discussed in [3, 25].

Example 1. For example $C = (\forall x(E(g(x), n) \vee L(x, n)) \wedge \forall x(E(x, n) \vee L(x, n))) \wedge \hat{Q}(n)$, P is a \mathcal{L}_F^ω sentence, where P consists of the following normalization rules:

$$\begin{aligned} \hat{Q}(0) &\Rightarrow \neg L(a, 0) \wedge \forall x(\neg E(x, 0) \vee \neg E(g(x), 0)) \\ \hat{Q}(s(n)) &\Rightarrow \forall x(\neg E(x, s(n)) \vee \neg E(g(x), s(n))) \forall x(\neg L(x, s(n)) \vee E(x, n) \vee L(x, n)) \\ &\quad \wedge \forall x(\neg L(g(x), s(n)) \vee E(g(x), n) \vee L(x, n)) \wedge \hat{Q}(n) \end{aligned}$$

and $\Sigma_l = \{a, g(\cdot)\}$. The normalization rules for the defined predicate symbol \hat{Q} are constructed from the following 0-abstract sentences: $\neg L(a, 0) \wedge \forall x(\neg E(x, 0) \vee \neg E(g(x), 0))$ and

$$\begin{aligned} &\forall x(\neg E(x, s(m)) \vee \neg E(g(x), s(m))) \forall x(\neg L(x, s(m)) \vee E(x, n) \vee L(x, m)) \\ &\quad \wedge \forall x(\neg L(g(x), s(m)) \vee E(g(x), m) \vee L(x, m)) \quad \wedge \bullet \end{aligned}$$

4 Evaluation of Abstract Sentences

Abstract sentences of \mathcal{L}_F^ω can be evaluated to static essential sentences by applying *ground substitutions* which replace the free ω variables by numerals. After applying such ground substitutions one can apply the normalization rules associated with the abstract sentence resulting in a sentence without defined symbols.

The domain of a ground substitution σ will be denoted by $dom(\sigma)$ and the set of all ground substitutions with domain V will be denoted by \mathcal{G}_V . Ground substitutions can be partially ordered with respect to their domains. We say that $\sigma \preceq_V \sigma'$ for ground substitutions $\sigma, \sigma' \in \mathcal{G}_V$, if for all $x \in V$, $x\sigma \leq x\sigma'$. We say that $\sigma \prec_V \sigma'$ if for at least one $x \in V$, $x\sigma < x\sigma'$. We can apply a ground substitution σ to a sentence $F \in \mathcal{L}_F^\omega$. The essential sentence resulting from application of a ground substitution $\sigma \in \mathcal{G}_{\mathcal{V}_\omega(F)}$ to an abstract sentence $F \in \mathcal{L}_F^\omega$ will be referred to as the *normalization of F by σ* . Note that the essential sentences resulting from normalization are *static*.

Theorem 1. *Let $C = (F, S) \in \mathcal{L}_F^\omega$ s.t. $\mathcal{V}_\omega(F) = V$ and σ a ground substitution s.t. $dom(\sigma) = V$. Then $N(F\sigma, S)$ is a static essential formula where $N(G, S')$ is defined as follows:*

$$\begin{aligned} N(P(\bar{t}, \bar{\alpha}), S') &\implies P(\bar{t}, \bar{\alpha}) & N(\neg P(\bar{t}, \bar{\alpha}), S') &\implies \neg P(\bar{t}, \bar{\alpha}) \\ N(\forall x Q(x), S') &\implies \forall x N(Q(x), S') & N(P \vee Q, S') &\implies N(P, S') \vee N(Q, S') \\ N(P \wedge Q, S') &\implies N(P, S') \wedge N(Q, S') & N(\hat{P}(s(t)), S') &\implies N(Q(n) \{n \leftarrow t\}, S') \\ N(\hat{P}(0), S') &\implies N(G, S') \end{aligned}$$

where $\hat{P}(s(n)) \implies Q(s(n))$, $\hat{P}(0) \implies G \in S'$.

Proof. Given that $\mathcal{L}_F^\omega \equiv \bigcup_{i=0}^\infty F_i$ and $F_i, S_i \subset F_{i+1}$, the theorem easily follows by induction on i , i.e. the abstraction complexity.

Example 2. The normalization of C by the ground substitution $\{n \leftarrow s^2(0)\}$ results in the following static essential sentence:

$$\begin{aligned} &\forall x (EQ(g(x), s^2(0)) \vee LE(x, s^2(0)) \wedge \forall x (EQ(x, s^2(0)) \vee LE(x, s^2(0))) \wedge \\ &\forall x (\neg LE(g(x), s^2(0)) \vee EQ(g(x), s(0)) \vee LE(x, s(0))) \wedge \\ &\forall x (\neg LE(x, s^2(0)) \vee EQ(x, s(0)) \vee LE(x, s(0))) \wedge \forall x (\neg EQ(x, s^2(0)) \vee \neg EQ(g(x), s^2(0))) \wedge \\ &\forall x (\neg LE(g(x), s(0)) \vee EQ(g(x), 0) \vee LE(x, 0)) \wedge \forall x (\neg LE(x, s(0)) \vee EQ(x, 0) \vee LE(x, 0)) \wedge \\ &\forall x (\neg EQ(x, s(0)) \vee \neg EQ(g(x), s(0))) \wedge \forall x (\neg EQ(x, 0) \vee \neg EQ(g(x), 0)) \wedge \neg LE(a, 0) \end{aligned}$$

Furthermore, we will represent the resulting static essential sentences in *conjunctive normal form* (maintaining logical equivalence), in other words, a sequence of *clauses*. A *clause* is defined as a disjunction of literals (i.e. P or $\neg P$ where P is an atomic formula). This implies that the result of normalizing a formula F by a substitution σ can be translated to a clause set which we denoted by $C(F, \sigma)$. Given an abstract sentence $F \in \mathcal{L}_F^\omega$, a clause c is said to be in F if there exists $\sigma \in \mathcal{G}_{\mathcal{V}_\omega(F)}$ such that $c \in C(F, \sigma)$. When an abstract sentence is normalized and placed in clausal form the quantifiers may be prenexified, i.e. only the individual clauses occur within the scope of a quantifier making syntactic representation of quantifiers superfluous.

5 Refutations, Associations, and Complexity

In general, the satisfiability problem for \mathcal{L}_F^ω is undecidable (follows from the undecidability of first-order satisfiability) as well as proving that a given $F \in \mathcal{L}_F^\omega$

is unsatisfiable for every instance. However, given a ground substitution σ s.t. $\text{dom}(\sigma) = \mathcal{V}_\omega(F)$, $F\sigma$ is a first-order sentence and thus standard theorem proving techniques may be applied, i.e. if $F\sigma$ is unsatisfiable there is a closed *semantic tree* constructible from $F\sigma$ [24].

In this work, we only consider $F \in \mathcal{L}_F^\omega$ which are unsatisfiable and focus on the representation problem, that is providing a finite representation of a proof that F is indeed unsatisfiability for every instance. Proof analysis in the presence of induction involves the construction of such finite representation. Global cut-elimination methods such as CERES [6] when used for proof analysis, extract the cut structure of a formal proof as a negation normal form formula (NNF) [25] or as a clause set [6] which is always unsatisfiable. Generalizations of the method [18, 25] which handle infinite sequences of formal proofs representing an inductive argument, extract the cut structure as a recursively define NNF formula. These formula, resulting from *schematic proof analysis*, are \mathcal{L}_F^ω formula.

However even for this restricted subset of \mathcal{L}_F^ω the representation problem is undecidable (a corollary of Rice's theorem [32]), though certain aspects of the full decision problem are simplified. For example, knowing that every instance is unsatisfiable allows one to take advantage of existing theorem proving technology, see [15] where instance analysis lead to a uniform finite representation. Furthermore, associations between various instances of the same abstract formula can be drawn allowing one to develop an alternative concept of complexity based on the occurrences of formula within a given proof of unsatisfiability.

In this section we develop the structures necessary for drawing these associations and defining a complexity measure based on these associations. Our complexity measure differs from than Herbrand sequent size or proof length in that it measures how associated clauses occur within the proofs of unsatisfiability for various instances of an abstract formula F . This allows us to avoid certain aspects of \mathcal{L}_F^ω which contribute to standard complexity measures but do not imply much about the inductive complexity of the sentence. For example, the complexity of the inductive propositional structure, a problem handled at length by V. Aravantinos *et al.*[2, 3].

5.1 Refutations: Proof of Unsatisfiability

First-order resolution [24] may be used to provide a proof of unsatisfiability for normalizations of $F \in \mathcal{L}_F^\omega$. The resolution refutations which result from the application of resolution theorem provers (such as SPASS [37] or Vampire [23]) to normalizations of $F \in \mathcal{L}_F^\omega$ can be translated into **LK**-calculus [35] as a particular type of **LK**-derivation. The leaves of these **LK**-derivations are instances of clauses, and only contain cut and contraction inferences. We will refer to them as **LK**-refutations. Note that such **LK**-derivations always end in the empty sequent and the unifications associated with resolution inferences are pushed to the leaves of the derivation. An implementation of this translation may be found in the GAP system [20].

Concerning the representation of clauses within the **LK**-refutations, we consider the classical interpretation of a sequent, namely $\left(\bigvee_{p \in \Pi} \neg p \vee \bigvee_{p \in \Delta} p \right) \equiv$

$\Pi \vdash \Delta$ where each $p \in \Pi \cup \Delta$ is atomic. For a given $F \in \mathcal{L}_F^\omega$ and ground substitution σ we say a clause $c \in C(F, \sigma)$ *occurs in a* **LK**-refutations R if there exists an initial sequent whose clausal form is an instance of c (modulo substitutions applied to variables of the ι sort). Furthermore, if a clause $c \in C(F, \sigma)$ occurs in a **LK**-refutations R we refer to the initial sequent which is an instance of c as an *occurrence of c in R* . By $\mathcal{R}(F, \sigma)$ we denote the **LK**-refutations of $C(F, \sigma)$.

5.2 Associations: Constructing Coverings

Let us consider some $F \in \mathcal{L}_F^\omega$ and two ground substitutions σ and σ' s.t. $\text{dom}(\sigma) = \text{dom}(\sigma') = \mathcal{V}_\omega(F)$ and $\sigma \leq_{\mathcal{V}_\omega(F)} \sigma'$. If we are to pick two refutations $R \in \mathcal{R}(F, \sigma)$ and $R' \in \mathcal{R}(F, \sigma')$ we can ask which clauses $c \in C(F, \sigma)$ occur in both R and R' . If such clauses do exist, this would provide insight into the relationship between the two refutations. This is a simple form of *association*.

Consider instead $c \in C(F, \sigma)$ which occurs in R but not in R' and a $c' \in C(F, \sigma')$ which does not occur in R but does occur in R' . Obviously these need not be occurrences of the same clause but may be instances of a more general pattern. A *cover* is a set of abstract formula which capture all instances of an $F \in \mathcal{L}_F^\omega$. In particular, we are interested in covers which are *finite* and *most general*. Covers are in some sense generalized clause sets which allow us to abstract away from certain aspects of a refutation.

In order to provide a proper definition of a cover, we need to introduce the concept of an ι -substitutions which map ι variables to variable-free ι -terms. Ground ι -substitutions map ι variables to variable-free ι -terms.

Definition 4. Let $F \in \mathcal{L}_F^\omega$, s.t. $\mathcal{V}_\omega(F) = V$, c_1, c_2 are clauses of F s.t. $c_1 \in C(F, \sigma)$, $c_2 \in C(F, \sigma')$, σ and σ' are ground ω -substitutions such that $\sigma \leq_V \sigma'$, and $\tau_\sigma, \tau'_\sigma, \tau_{\sigma'}$, and $\tau'_{\sigma'}$ are ι -substitutions dependent on the choice of σ and σ' . We refer to c_1 and c_2 as *siblings* if there exists a sentence $c \in \mathcal{L}_F^\omega$, where $\mathcal{V}_\omega(c) \subseteq V$ and $c\sigma\tau_\sigma = c_1\tau_\sigma$ and $c\sigma'\tau_{\sigma'} = c_2\tau'_{\sigma'}$. We refer to c as the *parent* of c_1 and c_2 . A parent can have more than two children. We denote the set of all clauses of F which are children of c by $\text{Ch}(c, F)$.

Notice that Definition 4 uses a concept similar to anti-unification [29, 31] to define the relation between siblings, though the concept of least general is closer to θ -subsumption [26, 30] in our case. Consider the clauses $C_1 \equiv \forall x(EQ(x, s^2(0)) \vee LE(x, s^2(0)))$ and $C_2 \equiv \forall x(EQ(x, s(0)) \vee LE(x, s(0)))$. Obviously, $C_1 \not\equiv C_2$, but they are indeed variations of the same clause modulo the chosen ground and ι -substitution, their parent being the clause $\forall x(EQ(x, n) \vee LE(x, n))$. Notice that there is an implicit ordering on parents based on their generality.

Definition 5. Let p, p' be parents of a clause c of F . We say p' is *more general* than p if $\text{Ch}(p, F) \subseteq \text{Ch}(p', F)$. This relationship will be denoted as $p \ll_{(c, F)} p'$. A parent p is said to be *most general* if for any clause $c \in \text{Ch}(p, F)$ and any p' s.t. $c \in \text{Ch}(p', F)$, $p' \ll_{(c, F)} p$.

A sentence F can be *covered* by a set of parents, that is a set of parents C covers F if for any ground substitution σ , and any clause $c \in C(\sigma, F)$ there exists a parent in $p \in C$ a ground substitution σ' , and two ι -substitutions $\tau_{\sigma'}$ and $\tau'_{\sigma'}$, s.t. $p\sigma'\tau_{\sigma'} = c\tau'_{\sigma'}$. A cover is *minimal* if every $p \in C$ is most general.

Example 3. Consider the formula constructed in Example 2, it has the following minimal cover which we refer to as $Cov(C)$:

$$(\neg L(x, s(n)) \vee E(x, n) \vee L(y, n)), (\neg E(x, n) \vee \neg E(g(x), n)), (E(x, n) \vee L(y, n)), \neg L(a, 0)$$

Lemma 1. *The clauses of Example 3 form a minimal cover of Example 2.*

Proof. Trivial, can be done by checking instantiations.

The cover of a sentence $F \in \mathcal{L}_F^\omega$ need not be finite nor need it consists of essential 0-abstract sentences only. Infinite covers can occur when a formula $G \in \mathcal{L}_F^\omega$ contains a clause whose most general parent is either an schematic 0-abstract sentence or an n-abstract sentence for $n > 0$. However, variable length clauses need not be so complex, for example consider the following:

Example 4. Consider, $NIA = (\forall x L(x, x) \wedge \hat{T}(n) \wedge \forall x(\hat{Q}(n, x)), S)$ where S contains the following normalization rules:

$$\begin{aligned} \hat{Q}(s(n), x) &\implies E(x, s(n)) \vee \hat{Q}(n, x) \\ \hat{Q}(0, x) &\implies E(x, 0) \\ \hat{T}(s(n)) &\implies \forall x, y, z (\neg L(m(x, y), z) \vee L(x, z) \wedge \\ &\quad \forall x, y, z (\neg L(m(x, y), z) \vee L(y, z) \wedge \\ &\quad \forall x, y (\neg L(g(y), x) \vee \neg E(y, s(n)) \vee \neg E(x, s(n)))) \wedge \hat{T}(n) \\ \hat{T}(0) &\implies \forall x, y (\neg L(g(y), x) \vee \neg E(y, 0) \vee \neg E(x, 0)) \end{aligned}$$

Notice that $\forall x E(x, 0)$, $\forall x (E(x, 0) \vee E(x, s(0)))$, and $\forall x (E(x, 0) \vee E(x, s(0)) \vee E(x, s(s(0))))$ are clauses of G which are of different size and shape, but are different instances of the same pattern. The most general parent of these three clauses is precisely $(\hat{Q}(n, x), S')$ where S' contains the normalization rules for \hat{Q} . The parent is not always found within the original abstract formula.

5.3 Complexity: Occurrences per Clause in Covering

Our notion of complexity concerns unsatisfiable sentences of \mathcal{L}_F^ω which have finite minimal coverings. While this does avoid a large portion of unsatisfiable sentences of \mathcal{L}_F^ω , restricting ourselves to finite minimal coverings, this allows one to perform proof analysis and use interactive theorem proving methods such as those mentioned in earlier work [15].

Let $F \in \mathcal{L}_F^\omega$, $Cov(F)$ its finite minimal cover, $c \in Cov(F)$, $\sigma \in \mathcal{G}_{\mathcal{V}_\omega(c)}$, and τ a ground ι -substitution τ with domain $bd(c\sigma)$. For every ground substitution $\sigma' \in \mathcal{G}_{\mathcal{V}_\omega(c)}$, s.t. $bd(c\sigma) = bd(c\sigma')$, $c\sigma\tau$ and $c\sigma'\tau$ are ι -equivalent, denoted by $c\sigma\tau \equiv_i c\sigma'\tau$, otherwise they will be referred to as ι -inequivalent. This concept is essentially counting instances of a clause $c \in Cov(F)$ occurring within a given

refutation. Changes in the numeric terms within with respect to a clause $c \in \text{Cov}(F)$ highlight the inductive propositional structure and not the inductive term structure. The latter is precisely what we want to capture.

Now we can define our counting mechanism. Let $\text{Cov}(F)$ be a finite minimal cover of a sentence $F \in \mathcal{L}_F^\omega$, $c \in \text{Cov}(F)$, and σ a ground substitution such that $\text{dom}(\sigma) = \mathcal{V}_\omega(F)$. We denote by $|R|_c$ for $R \in \mathcal{R}(F, \sigma)$ the number of pairwise ι -inequivalent occurrences of clauses $d \in \text{Ch}(c, F)$ in R . By \mathcal{S}_c^F we denote the set of all ground substitutions σ s.t. there exists $c' \in C(F, \sigma)$ and $c' \in \text{Ch}(c, F)$. A *chain of substitutions of \mathcal{S}_c^F* , is a total function $C : \mathbb{N} \rightarrow \mathcal{S}_c^F$ s.t. $C(n) <_{\mathcal{V}_\omega(F)} C(n+1)$.

Definition 6. Let $F \in \mathcal{L}_F^\omega$, $\text{Cov}(F)$ a finite minimal cover of F , and $f : G_{\mathcal{V}_\omega(F)} \rightarrow \mathbb{N}$ a total function. We say that $\text{Cov}(F)$ is asymptotically bounded by f from above, if for every $c \in \text{Cov}(F)$ and for every chain of \mathcal{S}_c^F there exists a sequence of refutations $R_i \in \mathcal{R}(F, C(i))$ such that

$$|R_n|_c \leq \mathcal{O}(f(C(n))) \text{ as } n \rightarrow \infty$$

Furthermore, we say that $\text{Cov}(F)$ is asymptotically bounded by f from below, if for every $c \in \text{Cov}(F)$, for every chain of \mathcal{S}_c^F , every sequence of refutations $R_i \in \mathcal{R}(F, C(i))$ such that

$$\Omega(f(C(n))) \leq |R_n|_c \text{ as } n \rightarrow \infty$$

If $\text{Cov}(F)$ is asymptotically bounded by f from below and from above than we say that $\text{Cov}(F)$ is asymptotically enumerated by f denoted by

$$|R_n|_c \approx \Theta(f(C(n))) \text{ as } n \rightarrow \infty$$

To some extent Definition 6 is a measurement of the Herbrand sequent complexity with respect to a sequence of refutations. However, our notion of complexity ignores the propositional structure captured by Herbrand sequent complexity and is thus more meaningful with respect to inductive theorem proving. It captures the combinatorial complexity of the term structure.

For example, two abstract sentences whose minimal Herbrand sequent complexity differs schematically (is dependent on the instantiations of the ω variables) may be equated by Definition 6. Let us consider $(\hat{O}(n, m), S)$ defined in Example 5 and $\alpha, \beta \in \Omega_G$, s.t. $\alpha \neq \beta$. The complexity of $(\hat{O}(n, \alpha), S)$ and $(\hat{O}(n, \beta), S)$, with respect to Herbrand sequent size ($\alpha \cdot n$ and $\beta \cdot n$ instantiations, respectively), will differ. However, both have the same complexity by the measure presented in Definition 6. Also, not only do they have the same complexity, neither will be in the same complexity class as *NIA* of Example 4 even though they have the same number of free ω variables.

The essential difference between $(\hat{O}(n, \alpha), S)$, $(\hat{O}(n, \beta), S)$, and *NIA*, which is captured by Definition 6, is the hardness of providing a finite representation of the term instantiations within a sequence of refutations. This seems to be more important to inductive complexity than other measures of complexity [19].

An interesting observation is the relationship between $\Theta(1)$ problems and existing work. For example, $\Theta(1)$ problems are structurally *propositional schemata*,

as studied in earlier work [1, 3, 12], and are otherwise non-inductive concerning inductive term complexity. One may imagine this class of formula as an inductive Bernays-Schnfinkel-Ramsey Class [21]. Consider, $C_{base} = (\forall x(EQ(x, s(n)) \vee LE(x, s(n))) \wedge \hat{Q}(s(n)), S)$, where S contains the normalization rules:

$$\begin{aligned} \hat{Q}(0) &\Longrightarrow \neg LE(a, 0) \wedge \forall x(\neg EQ(x, 0)) \\ \hat{Q}(s(n)) &\Longrightarrow \forall x(\neg EQ(x, s(n))) \wedge \forall x(\neg LE(x, s(n)) \vee EQ(x, n) \vee LE(x, n)) \wedge \hat{Q}(n) \end{aligned}$$

Example 1 is asymptotically bounded from above by a linear function, we prove this in the next section, while Example 4, which is discussed in previous work [13, 14], is asymptotically bounded from above by a quadratic function. We conjecture that these problems highlight important boundaries of current inductive theorem proving technology. A closely related abstract formula, which is also asymptotically bounded from above by a quadratic function, can be easily shown to be a ‘‘diagonalization’’ of Example 1, see Example 5.

Example 5. The formula $(\hat{O}(n, m), S)$ is a generalization of Example 1 where S contains the following normalization rules.

$$\begin{array}{ll} \hat{O}(0, m) \Longrightarrow \hat{M}(a, m) \wedge \hat{P}(0, m) & \hat{P}(0, m) \Longrightarrow \forall y(\hat{C}(y, 0, m)) \wedge \neg LE(a, 0) \\ \hat{O}(s(n), m) \Longrightarrow \forall z(\hat{D}(z, z, s(n), m)) \wedge \hat{P}(s(n), m) & \hat{P}(s(n), m) \Longrightarrow (\forall y(\hat{C}(y, s(n), m)) \wedge \\ \hat{C}(y, n, 0) \Longrightarrow \neg E(y, n) & \qquad \qquad \qquad \forall z(\hat{T}(z, z, n, m)) \wedge \hat{P}(n, m) \\ \hat{C}(y, n, s(m)) \Longrightarrow \neg E(y, n) \vee \hat{C}(h(y), n, m) & \hat{M}(y, 0) \Longrightarrow E(0, y) \\ \hat{T}(y, z, n, 0) \Longrightarrow \neg LE(y, s(n)) \vee E(y, n) \vee LE(z, n) & \hat{M}(y, s(m)) \Longrightarrow E(0, y) \wedge \hat{M}(h(y), m) \\ \hat{T}(y, z, n, s(m)) \Longrightarrow \neg LE(y, s(n)) \vee E(y, n) \vee & \hat{D}(y, z, n, 0) \Longrightarrow (E(n, y) \vee LE(z, n)) \\ LE(z, n) \wedge \hat{T}(h(y), z, n, m) & \hat{D}(y, z, n, s(m)) \Longrightarrow (E(n, y) \vee LE(z, n)) \\ & \qquad \qquad \qquad \wedge \hat{D}(h(y), z, n, m) \end{array}$$

Example 1 is equivalent to $(\hat{O}(n, s(0)), S)$.

Example 5 is obviously beyond the scope of the superposition calculus by Peltier *et al.* [3] being that it contains two independent numeric variables. Evidence points towards the problem being beyond the scope of existing tree grammar based methods [19] as well.

This leads use to the conjecture that existing methods are limited to problems of linear complexity, and that the superposition calculus by Peltier *et al.* is refutationally complete for this class of problems. Though, investigations into problems of complexity between quadratic and linear have not been performed.

In the following section we prove that the finite minimal cover of Example 1 is asymptotically enumerated by $\Theta(\alpha)$. We refrain from repeating the proofs concerning the asymptotic complexity of Example 4 [13, 14]. In these earlier works Example 4 was shown to be asymptotically bounded from above by $O(\alpha^2)$. However, it is unknown whether or not it is asymptotically bounded from below by $\Omega(\alpha^2)$. Example 4 is a variant of the infinitary pigeonhole principle, a problem which is known for its combinatorial difficulty. While by other complexity measures its hardness can be precisely stated we do not have tight lower bounds and can only conjecture its enumeration by $\Theta(\alpha^2)$. Example 5 is quite obviously of quadratic complex, this becomes apparent after our the analysis in Section 6 and the fact that it is the diagonalization of a sequence of problems of $\Theta(\alpha)$ complexity with increasing Herbrand sequent complexity.

6 A Linear Complexity Abstract Sentence

In this section we cover the asymptotic complexity of C introduced in Example 1, whose finite minimal cover $Cov(C)$ is given in Example 3. The fact that this cover is asymptotically enumerated by $\Theta(\alpha)$ can be shown by providing a necessary and sufficient sequence of instantiations of the bound variables.

Rather than providing the instantiations directly, we can instead express the instantiations through the minimal finite cover as done below:

$$\begin{aligned} \hat{C}_1(s(n), m, x) &\equiv \hat{C}_1(n, m, g(x)) & \hat{C}_1(0, m, x) &\equiv E(x, m) \vee L(x, m) \\ \hat{C}_2(s(n), m, x) &\equiv \hat{C}_2(n, m, g(x)) & \hat{C}_2(0, m, x) &\equiv \neg E(x, m) \vee \neg E(g(x), m) \\ \hat{C}_3(s(n), m, x) &\equiv \hat{C}_3(n, m, g(x)) & \hat{C}_3(0, m, x) &\equiv \neg L(x, s(m)) \vee E(x, m) \vee L(x, m) \end{aligned}$$

We will refer to this variant of the cover as

$$Cov_g(C) \equiv \left\{ \hat{C}_1(n, m, a), \hat{C}_2(n, m, a), \hat{C}_3(n, m, a), LE(a, 0) \right\}.$$

There exists instantiations of the clauses found $Cov(C)$ which cannot be constructed from the clauses of $Cov_g(C)$, but as we will show, such instantiations are not necessary for constructing a refutation of C . There exists a formulation of C , $C_g = (\hat{R}(n, n, a), S')$, where S' contains the following normalization rules:

$$\begin{aligned} \hat{R}(0, m, y) &\implies (EQ(g(y), s(n)) \vee LE(y, s(n)) \wedge (EQ(y, s(n)) \vee LE(y, s(n)))) \wedge \\ &\quad \hat{Q}(s(n), y) \\ \hat{R}(s(n), m, y) &\implies (EQ(g(y), s(n)) \vee LE(y, s(n)) \wedge (EQ(y, s(n)) \vee LE(y, s(n)))) \wedge \\ &\quad \hat{Q}(s(n), y) \wedge \hat{R}(n, m, g(y)) \\ \hat{Q}(0, y) &\implies \neg LE(a, 0) \wedge (\neg EQ(y, 0) \vee \neg EQ(g(y), 0)) \\ \hat{Q}(s(n), y) &\implies (\neg EQ(y, s(n)) \vee \neg EQ(g(y), s(n))) (\neg LE(y, s(n)) \vee EQ(y, n) \vee LE(y, n)) \\ &\quad \wedge (\neg LE(g(y), s(n)) \vee EQ(g(y), n) \vee LE(y, n)) \wedge \hat{Q}(n, y) \end{aligned}$$

Which only contains terms which are covered by $Cov_g(C)$.

Lemma 2. $Cov_g(C)$ is a finite cover of C_g .

Proof. This follows from the definition of \hat{R} .

We show that a particular subset of the clauses of C_g is enough to refute C_g and while a smaller subset might also provide a refutation, it will use precisely the same terms of the ι sort, i.e. asymptotically bounded from below. Note, we do not cover the case when $\alpha = 0$ in Theorem 2 because it is trivial and does not use the induction invariant covered in the proof.

Theorem 2. Let $s(0) \leq \alpha$ be a numeral. Then there exists a refutation of C_g using the following subformula labeled by clause occurrences:

$$\begin{aligned} &\left(\bigwedge_{i=0}^{\alpha} \hat{C}_1(i, \alpha, a) \wedge \hat{C}_1(s(i), \alpha, a) \right) \wedge \left(\bigwedge_{i=0}^{\alpha} \bigwedge_{j=0}^i \hat{C}_2(j, i, a) \right) \wedge \neg LE(a, 0) \wedge \\ &\quad \left(\bigwedge_{i=0}^{\alpha} \bigwedge_{j=0}^{i-1} \hat{C}_3(j, i-1, a) \wedge \hat{C}_3(s(j), i-1, a) \right) \end{aligned} \tag{1}$$

Where $\bigwedge_{j=0}^{0-1} \hat{C}_3(j, i-1, a) \wedge \hat{C}_3(s(j), i-1, a) \equiv \top$.

Proof. Let us consider the basecase when $\alpha = s(0)$, in particular we can first resolve $\hat{C}_1(s(0), s(0), a)$ and $\hat{C}_1(s(s(0)), s(0), a)$ with $\hat{C}_2(s(0), s(0), a)$ and then apply contraction. The result is the clause $L(g(a), s(0))$. Resolving $L(g(a), s(0))$ with $\hat{C}_3(0, s(0), a)$ results in $\hat{C}_1(0, 0, a)$. Furthermore, resolving $\hat{C}_1(0, s(0), a)$ and $\hat{C}_1(s(0), s(0), a)$ with $\hat{C}_2(0, s(0), a)$ results in the clause $LE(a, s(0))$ after contraction. Resolving $L(a, s(0))$ with $\hat{C}_4(s(0), s(0), a)$ results in $\hat{C}_1(s(0), 0, a)$. From $\hat{C}_1(0, 0, a)$ and $\hat{C}_1(s(0), 0, a)$ we can easily arrive at a contradiction by resolving $\neg L(a, 0)$ with both $\hat{C}_1(0, 0, a)$ and $\hat{C}_1(s(0), 0, a)$, resulting in $E(g(a), 0)$ and $E(a, 0)$ which can finally be resolved with $\hat{C}_2(0, 0, a)$.

Let us assume that the statement holds for all numerals $n \leq \alpha$ and show that the statement holds for $s(\alpha)$. Following the procedure outlined in the base case we show that one can derive Equation 2b from Equation 2a using Equation 3a and Equation 3b.

$$\bigwedge_{i=0}^{s(\alpha)} \hat{C}_1(i, s(\alpha), a) \wedge \hat{C}_1(s(i), s(\alpha), a) \quad (2a) \qquad \bigwedge_{j=0}^{s(\alpha)} \hat{C}_2(j, s(\alpha), a) \quad (3a)$$

$$\bigwedge_{i=0}^{\alpha} \hat{C}_1(i, \alpha, a) \wedge \hat{C}_1(s(i), \alpha, a) \quad (2b) \qquad \bigwedge_{j=0}^{\alpha} \hat{C}_3(j, \alpha, a) \wedge \hat{C}_3(s(j), \alpha, a) \quad (3b)$$

By resolving (2a) with (3a) we can derive the sequence

$$\bigwedge_{i=0}^{\alpha} L(g^i(a), s(\alpha)) \quad (4)$$

where $g^i(a)$ is an abbreviation for $\overbrace{g(g(\dots g(a))\dots)}^{i \text{ times}}$. Resolving (4) with (3b) results in Equation 2b. Together with the clauses

$$\bigwedge_{i=0}^{\alpha} \left(\left(\bigwedge_{j=0}^i \hat{C}_2(j, i, a) \right) \wedge \left(\bigwedge_{j=0}^{i-1} \hat{C}_3(s(j), i-1, a) \wedge \hat{C}_3(s(j), i-1, a) \right) \right) \wedge \neg LE(a, 0)$$

we have proven the stepcase. \square

A corollary of this result is as follows:

Corollary 1. *The minimal finite cover $Cov(C)$ is asymptotically bounded by $O(\alpha)$ from above.*

Proof (sketch). Theorem 2 provides a sequence of refutations of \mathcal{C} . These refutations can be ordered with respect to the chain of substitutions $C'(\alpha) = \{n \leftarrow \alpha\}$, that is R_α is the refutation corresponding to the clause set $C(C, C'(\alpha))$. Consider the clause $\neg L(x, s(n)) \vee E(x, n) \vee L(x, n)$ from $Cov(C)$, there are precisely $\left(\frac{2 \cdot \alpha \cdot (2 \cdot \alpha + 1)}{2} - 1\right)$ occurrences of this clause in $R_{s(\alpha)}$, but invariant the terms of the ω sort there are only α occurrences. Thus the cover is asymptotically bounded by $O(\alpha)$. \square

The next step is to show that the minimal finite cover $Cov(C)$ is asymptotically bounded by α from below. Unlike showing that a cover is asymptotically bounded from above, which merely asked for a witnessing sequence of refutations, one must show that no refutation sequence exists with a lower asymptotic complexity. However, given the simplicity of the clause set C we can use a *semantic tree* construction [24] to provide the non-existence of a refutation sequence with a lower asymptotic complexity. First consider the Herbrand universe of C with respect to the ι sort, namely, $H = \{a, g(a), g(g(a)), \dots\}$. The refutation sequence provided in Theorem 2 uses the subset $H' = \{a, \dots, g^\alpha(a)\}$ to refute $C \{n \leftarrow \alpha\}$. Essentially to show that there does not exist a refutation sequence with a lower asymptotic complexity we must show that no subset of $G \subset \{a, \dots, g^\alpha(a)\}$ is sufficient for refuting $C \{n \leftarrow \alpha\}$ or that any semantic tree for $C \{n \leftarrow \alpha\}$ constructed from $G \cup (H \setminus H')$ will have an open branch.

Theorem 3. *Let $s(0) \leq \alpha$. Then any semantic tree for $C \{n \leftarrow \alpha\}$ constructed from $G \cup (H \setminus H')$ will have an open branch.*

Proof. We proceed by induction on α . When $\alpha = s(0)$ we may assume, without loss of generality, that $G = \{a\}$. Furthermore, we assume that a closed semantic tree can be constructed. Let us consider any branch containing $\neg L(a, 0)$. There are only two clauses which can close such a branch:

- a) $\neg L(g(a), s(0)) \vee E(g(a), 0) \vee L(a, 0)$
- b) $\neg L(a, s(0)) \vee E(a, 0) \vee L(a, 0)$

Case (a) contains $g(a)$ and thus is excluded by assumption. Concerning case (b), it implies the existence of a branch b containing $\neg L(a, 0)$, $L(a, s(0))$, and $\neg E(a, 0)$. If we limit ourselves to the terms $\{a\}$ then branch b cannot be closed, thus contradicting our assumption. If we allow literals to contain terms from $H \setminus \{a, g(a)\}$, then there are extensions of b which are closed. However, these extensions imply the existence of a branch which only contains negative occurrences of the symbol E and positive occurrences of L . No clauses can close this extension of the branch and thus such a branch contradicts our assumptions.

Now for the step case, without loss of generality we may assume that

$$G = \{a, \dots, g^{i-1}(a), g^{i+1}(a), \dots, g^{s(\alpha)}(a)\}.$$

Assume that for $C \{n \leftarrow \alpha\}$ any semantic tree S constructed from $G \cup H \setminus H'$ will have an open branch. We use this hypothesis to prove that any semantic tree S' constructed from $G \cup H \setminus H'$ for $C \{n \leftarrow s(\alpha)\}$ will have an open branch.

Let us consider the branch b of S closed by the clause $L(g^\beta(a), \alpha) \vee E(g^\beta(a), \alpha)$ where $\beta \neq i$. Now consider the closely related branch b' which contains the literals $L(g^\beta(a), \alpha)$ and $\neg E(g^\beta(a), \alpha)$, and thus cannot be closed by $L(g^\beta(a), \alpha) \vee E(g^\beta(a), \alpha)$. We may construct b' in such a way that it remains open. Extending b' by $\neg E(x, s(\alpha))$ and $L(x, s(\alpha))$ for $x \in G \cup H \setminus H'$ indefinitely can be done without closing the branch, thus, proving the theorem.

Corollary 2. *The minimal finite cover $Cov(C)$ is asymptotically enumerated by α .*

Proof. Follows from Theorem 2 & 3.

Given the simplicity of Example 1, it, in some sense, provides a canonical example of what it means to be enumerated by a linear function. Furthermore, if our conjecture concerning the expressive power of certain inductive theorem provers holds, Example 1 can be seen as a benchmark for testing applicability of a theorem proving method to a given problem. As with our proposed proof of the quadratic enumeration of Example 4, showing that there is a bijection between the refutations of Example 1 and the refutations of a given abstract sentence provides evidence that certain inductive theorem provers can, in principal, provide an inductive invariant. Though, it is not clear if any prover is actually complete with respect to linearly enumerated abstract sentences. A good example of this issue is the *eventually constant assertion* presented in [15]. Drawing a bijection between it's and Example 1's refutations is quite simple, though an implementation of the superposition prover of Peltier *et al.*, which we conjecture to be linear complete, struggles to find an invariant in reasonable time.

7 Conclusion

In this paper we present a complexity measure for a particular type of primitive recursively defined unsatisfiable Σ_1 -sentences relevant for proof analysis and inductive theorem proving. The complexity measure is based on term tuple occurrences and the recursive difficulty of constructing the set of term tuple necessary for producing a refutation of a given sentence. We provide a few examples of essential classes in the complexity hierarchy constructed by the measure and prove that a variant of the infinitary pigeonhole principle is of linear complexity. This variant is part of a sequence of sentences whose limit is of quadratic complexity and is provided in Example 5.

In future work, we plan to investigate, in more depth, what problems fall into the various levels of our hierarchy. Especially of interest is the existence of sub-linear and sub-quadratic complexity sentences which are not of constant complexity or linear complexity. Furthermore, we plan to instigate the veracity of our conjecture that Peltier *et al.* super position prover is linear complete and whether or not this conjecture holds for other provers. Also, of interest is the discovery of a hierarchy of sentences whose complexity is an arbitrarily high polynomial degree. Towards this goal, We are currently investigating the combinatorics of *superstrings* [27] and how it relates to our measure. Note that problems discussed here are special cases of the superstring problem and have already found a connection between the superpermutation problem[28] and a so call super pigeonhole principle.

References

1. Vincent Aravantinos, Ricardo Caferra, and Nicolas Peltier. A schemata calculus for propositional logic. In *Proceedings of Tableaux '09*, volume 5607 of *lecture notes in com. sci.*, pages 32–46. Springer Berlin Heidelberg, 2009.
2. Vincent Aravantinos, Ricardo Caferra, and Nicolas Peltier. Decidability and undecidability results for propositional schemata. *Journal of Artificial Intelligence Research*, 40(1):599–656, January 2011.
3. Vincent Aravantinos, Mnacho Echenim, and Nicolas Peltier. A resolution calculus for first-order schemata. *Fundamenta Informaticae*, pages 101–133, 2013.
4. Matthias Baaz. Note on the generalization of calculations. *Theoretical Computer Science*, 224(12):3 – 11, 1999.
5. Matthias Baaz, Stefan Hetzl, Alexander Leitsch, Clemens Richter, and Hendrik Spohr. Ceres: An analysis of Fürstenberg’s proof of the infinity of primes. *Theoretical Computer Science*, 403(2-3):160–175, August 2008.
6. Matthias Baaz and Alexander Leitsch. Cut-elimination and redundancy-elimination by resolution. *J. of Symbolic Comput.*, 29:149–176, 2000.
7. Paul Beame and Toniann Pitassi. Current trends in theoretical computer science. chapter Propositional Proof Complexity: Past, Present, and Future, pages 42–70. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2001.
8. Robert S. Boyer and J. Strother Moore. A theorem prover for a computational logic. In Mark E. Stickel, editor, *10th International Conference on Automated Deduction*, pages 1–15, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg.
9. James Brotherston. Cyclic proofs for first-order logic with inductive definitions. In Bernhard Beckert, editor, *Proceeding of TABLEAUX '05*, volume 3702 of *lecture notes in com. sci.*, pages 78–92. Springer Berlin Heidelberg, 2005.
10. James Brotherston, Nikos Gorogiannis, and Rasmus L. Petersen. A generic cyclic theorem prover. In Ranjit Jhala and Atsushi Igarashi, editors, *Programming Languages and Systems*, pages 350–367, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
11. James Brotherston and Alex Simpson. Sequent calculi for induction and infinite descent. *Journal of Logic and Computation*, pages 1177–1216, 2010.
12. David Cerna. A tableaux-based decision procedure for multi-parameter propositional schemata. In StephenM. Watt, JamesH. Davenport, AlanP. Sexton, Petr Sojka, and Josef Urban, editors, *Intelligent Computer Mathematics*, volume 8543 of *lecture notes in com. sci.*, pages 61–75. Springer International Publishing, 2014. Peer Reviewed.
13. David M. Cerna. *Advances in schematic cut elimination*. PhD thesis, Technical University of Vienna, 2015. Dissertation: urn:nbn:at:at:ubtuw:1-84124.
14. David M. Cerna. A General Recursive Construction for Schematic Resolution Derivations. Risc report series, Research Institute for Symbolic Computation (RISC), Johannes Kepler University Linz, Schloss Hagenberg, 4232 Hagenberg, Austria, 2018. submitted for review.
15. David M. Cerna and Alexander Leitsch. 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*, pages 241–256, 2016. Peer Reviewed.
16. Koen Claessen, Moa Johansson, Dan Rosén, and Nicholas Smallbone. Automating inductive proofs using theory exploration. In Maria Paola Bonacina, editor, *Automated Deduction – CADE-24*, pages 392–406, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

17. Stephen Cook and Phuong Nguyen. *Logical Foundations of Proof Complexity*. Cambridge University Press, New York, NY, USA, 1st edition, 2010.
18. Cvetan Dunchev, Alexander Leitsch, Mikheil Rukhaia, and Daniel Weller. Cut-elimination and proof schemata. In *Logic, Language, and Computation - 10th International Tbilisi Symposium on Logic, Language, and Computation, Tbilisi 2013, Gudauri, Georgia, September 23-27, 2013. Revised Selected Papers*, pages 117–136, 2013.
19. Sebastian Eberhard and Stefan Hetzl. Inductive theorem proving based on tree grammars. *Ann. Pure Appl. Logic*, 166(6):665–700, 2015.
20. Gabriel Ebner, Stefan Hetzl, Giselle Reis, Martin Riemer, Simon Wolfsteiner, and Sebastian Zivota. *System Description: GAPT 2.0*, pages 293–301. Springer International Publishing, Cham, 2016.
21. Egon Börger, Erich Grädel, and Yuri Gurevich. The classical decision problem (perspectives in mathematical logic). *Bulletin of the London Mathematical Society*, 30:317–335, 5 1998.
22. Donald E. Knuth. Big omicron and big omega and big theta. *SIGACT News*, 8(2):18–24, April 1976.
23. Laura Kovcs and Andrei Voronkov. First-order theorem proving and vampire. In Natasha Sharygina and Helmut Veith, editors, *Computer Aided Verification*, volume 8044 of *lecture notes in com. sci.*, pages 1–35. Springer Berlin Heidelberg, 2013.
24. Alexander Leitsch. The resolution calculus. *Journal of Logic, Language and Information*, 7(4):499–502, October 1998.
25. Alexander Leitsch, Nicolas Peltier, and Daniel Weller. Ceres for First-Order Schemata. *Journal of Logic and Computation*, 2017. To appear.
26. Jérôme Maloberti and Michèle Sebag. Fast theta-subsumption with constraint satisfaction algorithms. *Machine Learning*, 55(2):137–174, May 2004.
27. Martin Middendorf. More on the complexity of common superstring and supersequence problems. *Theor. Comput. Sci.*, 125(2):205–228, 1994.
28. OEIS. A180632, 2010.
29. G. D. Plotkin. A note on inductive generalization. In *Machine Intelligence 5*, pages 153–163. Elsevier, 1970.
30. G. D. Plotkin. A further note on inductive generalization. In *Machine Intelligence 6*, pages 100–124. Elsevier, 1971.
31. J. C. Reynolds. Transformational systems and the algebraic structure of atomic formulas. In *Machine Intelligence 5*, pages 135–151. American Elsevier, 1970.
32. H. G. Rice. Classes of recursively enumerable sets and their decision problems. *Transactions of the American Mathematical Society*, 74(2):358–366, 1953.
33. William Sonnex, Sophia Drossopoulou, and Susan Eisenbach. Zeno: An automated prover for properties of recursive data structures. pages 407–421. Springer Berlin Heidelberg, 2012.
34. G. Sutcliffe. The TPTP Problem Library and Associated Infrastructure. From CNF to TH0, TPTP v6.4.0. *Journal of Automated Reasoning*, 59(4):483–502, 2017.
35. Gaisi Takeuti. *Proof Theory*, volume 81 of *Studies in logic and the foundations of mathematics*. American Elsevier Pub., 1975.
36. Jannik Vierling. Cyclic superposition and induction. Master’s thesis, Technical University of Vienna, 2018.
37. Christoph Weidenbach, Dilyana Dimova, Arnaud Fietzke, Rohit Kumar, Martin Suda, and Patrick Wischniewski. Spass version 3.5. In *Proceedings of the 22Nd International Conference on Automated Deduction, CADE-22*, pages 140–145. Springer-Verlag, 2009.