TECHNISCHE
UNIVERSITÄT
WIEN

DISSERTATION

The limits of automated inductive theorem provers

Ausgeführt zum Zwecke der Erlangung des akademischen Grades
eines Doktors der technischen Wissenschaften unter der Leitung
von

Assoc. Prof. Dr. techn. Stefan Hetzl

eingereicht an der Fakultät für Mathematik und Geoinformation
der Technischen Universität Wien

von

Dipl.-Ing. Jannik Vierling BSc.

Acknowledgments

First and foremost, I would like to thank my advisor Stefan Hetzl for giving me the chance to work as a researcher in the field of computational logic. I am particularly, grateful for his guidance and his invaluable advice that let me become a better researcher. Without Stefan's support this thesis would never have come to an end.

Furthermore, I would like to thank Pavel Pudlák for hosting me three months in Prague. During this stay I had the opportunity to discuss my work with Emil Jeřabek whose suggestions helped me to prove a central result of this thesis. Moreover, I would like to thank Lev Beklemishev for inviting me to present my work at the Steklov institute. My thanks also go to Christoph Weidenbach who hosted me in Saarbrücken and let me share my work with his group. I am also very grateful to Matthias Baaz and Laura Kovács for their help with the funding of this thesis.

I would like to thank my parents for encouraging me in developing a scientific mindset, and thus providing the foundations for my work on this thesis. Finally, special thanks go to my wife, Petra, for her love and support during all these years, despite my frequent absent-mindedness.

The work on this thesis was funded by WWTF VRG 12-004, FWF I 4427, and ERC CoG ARTIST 101002685.

Abstract

In this thesis we formally analyze the limits of several saturation-based inductive theorem provers. We propose an analysis technique that consists in reducing provers into first-order theories with induction. At first sight the reduction of a prover into a first-order theory may seem to be a very strong abstraction. Therefore, it is a priori not clear whether such reductions retain some useful information about the original prover. In this thesis we show that this approach permits to extract crucial logical features and provides strong bounds on the logical strength of provers. Moreover, based on these bounds, we prove various unprovability results, whereas previously only empirical observations could be made based on the failure of concrete implementations. The unprovability results in this thesis show that, despite the loss of details incurred by the reduction of a prover to a first-order theory, there are elementary properties that are not provable by recent automated inductive theorem provers even given any amount of time and memory. Furthermore, the thesis links these unprovability results to a certain extent to the logical features of the provers and thus provides some guidance for the development of more powerful automated inductive theorem provers.

Contents

1 Introduction

Automated inductive theorem proving (AITP) is a branch of automated deduction that aims at automating the process of proving statements that involve inductively constructed objects such as natural numbers, finite sequences, and trees. In first-order automated theorem proving (ATP) we try to establish validity whereas in automated inductive theorem proving (AITP) one is usually interested in proving that a formula is true in the standard model of some inductive types. By Gödel's incompleteness theorem this problem is not semi-decidable. Since AITP systems are effective computational devices they only address semi-decidable subsets of this problem. Because of this discrepancy between semantics and effectiveness there is a lot of freedom in the choice of proof systems. In practice we see methods that make use of typical first-order induction schemata, Hilbert-style induction rules (for example [KP13; Ker14]), and cyclic calculi (see [Bro05; BGP12]) that exceed the power of the first-order induction schema [BT17; BT19]. The most prominent applications of automated inductive theorem proving are found in formal methods for software engineering. For example, the formal verification of software relies strongly on one or another form of induction, since any non-trivial program contains some form of loops or recursion. Another field of application of automated inductive theorem proving is the formalization of mathematical statements, where AITP systems assist humans in formalizing statements by discharging lemmas automatically, suggest inductions [Nag19], or explore the theory [JDB09; Joh+14; VJ15].

A large variety of methods for automating mathematical induction has been developed. Methods usually differ in the type of induction formulas they generate, the calculus they are integrated in, and other more technical features such as the degree of automation, the encoding of the input, semantics of datatypes, and so on. There are, for example, methods based on: recursion analysis [BM79; Ste88; Bun+89], proof by consistency [Com01], rippling [Bun+93], cyclic proofs [BGP12], extensions of saturation-based provers [Dar68; Biu+86; KP13; Ker14; Cru15; Cru17; EP20; RV19; Haj+20; Wan17], tree grammar provers [EH15], theory exploration based provers [Cla+13], rewriting induction [Red90], encoding [Sch20], extensions of SMT solvers [RK15]. Many methods integrate the induction mechanism more or less tightly into a proof system that is well-suited for efficient automation. Thus, these methods exist mainly at lower levels of abstraction, often close to an actual implementation. Therefore, the current methodology in automated inductive theorem proving focuses on empirical evaluations of its methods using sets of benchmark problems such as the ones described in [Cla+15; Haj+21a]. Such an empirical evaluation provides evidence about the strengths and weaknesses of a method but does not result in a systematic understanding of the underlying principles. However, formal explanations backing the observations obtained by the empirical evaluation are still rare. Hence, it is, as of now, difficult to classify methods according to their strength and to

give a theoretical explanation of an empirically observed failure of a given method in a particular context.

The work in this thesis is part of a research program addressing this situation by analyzing AITP systems using techniques and results from mathematical logic. Formal analyses of AITP systems complement and explain the knowledge obtained by the empirical analyses of these systems. Furthermore, formal analyses of AITP systems gives insights into the limits of the systems. Exploring the limits of AITP systems is particularly important because it gives us some information about potential applications and extensions of the systems and moreover allows us to obtain practically meaningful unprovability results where previously only empirical observations could be made based on the failure of a concrete system. Overall a formal analysis results in a better understanding of the logical composition of AITP systems and their relative strength. Thus this research program inevitably leads to a development of the logical foundations of automated inductive theorem proving.

This thesis develops a technique for the analysis of the limits of AITP systems and applies it to several concrete AITP systems. The analysis technique developed in this thesis consists in simulating AITP systems in suitable first-order theories. By simulating an AITP system in a first-order theory we obtain as a byproduct a relative soundness proof for the system but more importantly we reduce the provability in the concrete system to the provability in a first-order theory. Therefore, by Gödel's completeness theorem for first-order logic we can establish unprovability results by constructing suitable first-order models. Furthermore, the reduction of an AITP system to a first-order theory allows us to work on a highly abstract level that is mostly independent of details of the AITP systems. The main insight of the thesis is that despite of the high level of abstraction it is possible to obtain practically meaningful unprovability results for elementary properties about natural numbers. In particular, the technique permits the isolation of the features of an AITP system, that are limiting its overall expressive power.

The thesis mostly focuses on applying this technique to systems that extend a saturation-based proof system by an induction mechanism. Saturation-based theorem proving is currently the dominating paradigm in first-order theorem proving. Because of the recent advances in this paradigm, the subject of AITP has recently increasingly focused on integrating mathematical induction into saturation-based theorem provers [KP13; Ker14; Cru15; Cru17; Wan17; EP20; RV19; Haj+20]. Integrating induction into a saturation-based theorem prover involves many technical challenges that are due to the interaction of the induction mechanism with internal mechanisms of the prover such as Skolemization, clausification, clause splitting, and so on. In this thesis we will focus on the three systems: the system described in [RV19; Haj+20] and implemented in the Vampire theorem prover [KV13]; the system described in [Cru17] which is implemented in the theorem proving framework Zipperposition; and the system described in [KP13; Ker14] which is implemented as an extension of Prover9 [McC10].

The thesis is structured as follows. In Chapter 2 we will introduce some basic notions and results related to first-order logic, Skolemization, and induction that we will use throughout the thesis. In particular, we will define the inductive semantics that define the scope of the task of automated inductive theorem provers. In Chapter 3 we will

consider abstractly an explicit integration of induction into saturation and the interaction between induction and saturation. The main insight of this chapter is that Skolem symbols appearing in ground terms of induction formulas play the role of the parameters of induction axioms. This phenomenon occurs in the concrete systems we consider in Chapters 4 and 5. Chapter 3 is based on Sections 4 and 5 of [HV23]. In Chapter 4 we consider several variants of the induction rules described in [RV19; Haj+20; Haj+21b] and provide various unprovability results. Chapter 4 is a considerable extension of Section 6 of [HV23] in the sense that it also considers rules described in [Haj+21b] and provides many new unprovability results. In Chapter 5 we consider the AITP system described in [Cru17] which is following the architecture considered in Chapter 3 and moreover provides a lemma rule as well as explicit clause splitting. The highlight of this Chapter is a logical characterization of the refutational power of the considered AITP system which shows that this system subsumes all the systems considered in Chapter 4. Moreover, we show that this system is unable to prove the totality of certain simple function definitions. Chapter 6 considers systems based on clause set cycles which are an induction mechanism for saturation-based proof system based on the detection of a cyclic dependency. This chapter is based on the articles [HV20; HV22] which introduce clause set cycles as an abstraction of the cycles detected by the n-clause calculus [KP13; Ker14]. The highlights of this chapter are a logical characterization of the refutational power of clause set cycles as well the identification of several independent logical features that are responsible for various unprovability phenomena. Chapter 7 is dedicated to the proof of an independence result in linear arithmetic over natural numbers that underlies one of the unprovability results for clause set cycles given in Chapter 6. Finally, in Chapter 8 we briefly consider an entirely different family of AITP systems that are based on a theory-exploration approach [Cla+13]. Systems belonging to that family exhibit promising empirical behavior [Cla+13]. By making use of the technique developed in Chapters 4 to 6 we obtain a very coarse upper bound that nevertheless suffices to provide elementary unprovability results.

2 Preliminaries

In this section we introduce the formalism and related notions that we use in this thesis. In Section 2.1 we recall the basic notions of first-order logic and describe our notations. Section 2.2 introduces our variant of Skolemization and recalls some well-known results. We will in particular describe the naming schema for Skolem symbols adopted in this thesis. In Section 2.3 we define a sound and complete sequent calculus for first-order logic. In Section 2.4 we define the notions of literals, clauses and saturation systems as used in this thesis. After that, we define in Section 2.6 some simple theories of formal arithmetic that will be of use at various occasions. Finally, in Section 2.5 we define the inductive semantics and consider two effective systems for induction that we will use to provide upper bounds for the methods considered in this thesis.

2.1 First-order logic

We work in a setting of classical single-sorted first-order logic with equality. That is, besides the usual logical symbols we have a logical binary predicate symbol $=$ denoting equality. A first-order language $\mathcal{L}$ is a set of pairs of the form S/n where S is either a function symbol or a predicate symbol and $n \in \mathbb{N}$ denotes the arity of the symbol S. If the arity of a symbol is clear from the context we often simply write S in places where S/n is expected. Terms, atoms, and formulas are constructed as usual from function symbols, a fixed countable set $\mathfrak{V}$ of variable symbols, the logical connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, and the quantifiers $\exists$ and $\forall$. The set of all $\mathcal{L}$ formulas is denoted by $\mathcal{F}(\mathcal{L})$. The notions of bound variables and free variables are defined as usual. By $\mathrm{FV}(\varphi)$ we denote the set of free variables of a formula φ. A formula that has no free variables is called a sentence. Let t, s be terms, then by $t \neq s$ we abbreviate the formula $\neg t = s$. Let φ be a formula, then by $(\exists!y)\varphi(\vec{x}, y)$ we abbreviate the formula

$$(\exists y)\varphi(\vec{x}, y) \wedge (\forall y_1)(\forall y_2)(\varphi(\vec{x}, y_1) \wedge \varphi(\vec{x}, y_2) \rightarrow y_1 = y_2).$$

Let $x_1, \ldots, x_n$ be variables, $t_1, \ldots, t_n$ a terms, and φ a formula, then $\varphi[x_1/t_1, \ldots, x_n/t_n]$ denotes the simultaneous substitution of x_i by t_i for $i = 1, \ldots, n$ in φ. Let φ be a formula, then we write $\varphi(x_1, \ldots, x_n)$ to indicate that the free variables of φ are among $\{x_1, \ldots, x_n\}$. Let now $t_1, \ldots, t_n$ be terms, then we write $\varphi(t_1, \ldots, t_n)$ to denote the formula $\varphi[x_1/t_1, \ldots, x_n/t_n]$. We use analogous notations for terms.

In this thesis we will often be interested in the formulas that have a certain quantifier complexity.

Definition 2.1.1. *We say that a formula is $\exists_0$ (or $\forall_0$ or open) if it is quantifier-free. We say that a formula is $\exists_{n+1}$ ($\forall_{n+1}$) if it is of the form $\exists\vec{x}\varphi(\vec{x}, \vec{y})$, where φ is $\forall_n$ ($\exists_n$).*

Let $\mathcal{L}$ be a first-order language, by $\mathrm{Open}(\mathcal{L})$, $\exists_n(\mathcal{L})$, *and* $\forall_n(\mathcal{L})$ *we denote the set of open formulas,* $\exists_n$ *formulas, and* $\forall_n$ *formulas of the language* $\mathcal{L}$. *We say that a theory is* $\forall_n$ *(*$\exists_n$*) if all of its axioms are* $\forall_n$ *(*$\exists_n$*).*

We will often be more interested in the axioms of a theory, rather than the deductive closure of these axioms. Hence, we define a theory as a set axioms and manipulate the deductive closure by means of the first-order provability relation (see Definition 2.1.3).

Definition 2.1.2 (Theories)**.** *Let $\mathcal{L}$ be a first-order language, then a first-order $\mathcal{L}$ theory T is a set of $\mathcal{L}$ sentences called the axioms of T. Let T, U be theories, then $T+U$ denotes the theory axiomatized by $T \cup U$.*

Definition 2.1.3 (Provability)**.** *Let φ be a formula, then we write $T \vdash \varphi$ to denote that φ is provable in first-order logic from the axioms of T. We usually write $\vdash \varphi$ instead of $\varnothing \vdash \varphi$. Let Φ be a set of formulas, then we write $T \vdash \Phi$ to denote that $T \vdash \varphi$ for all formulas $\varphi \in \Phi$. Let T_1 and T_2 be theories, then we write $T_1 \equiv T_2$ if $T_1 \vdash T_2$ and $T_2 \vdash T_1$. We say that the theory T_2 is an extension of T_1 if $T_2 \vdash T_1$.*

Let T be a theory and φ a formula, then we write $T+\varphi$ to denote the theory axiomatized by the axioms of T and the universal closure of φ. We say that two formulas φ and ψ are logically equivalent if $\vdash \varphi \leftrightarrow \psi$.

Definition 2.1.4 (Conservativity)**.** *Let T_1 and T_2 be theories, and Φ a set of formulas. We say that T_1 is Φ-conservative over T_2 (in symbols $T_1 \sqsubseteq_\Phi T_2$), if, for all $\varphi \in \Phi$, $T_1 \vdash \varphi$ implies $T_2 \vdash \varphi$. We write $T_1 \equiv_\Phi T_2$ if $T_1 \sqsubseteq_\Phi T_2$ and $T_1 \sqsupseteq_\Phi T_2$. If $\Phi = \mathcal{F}(\mathcal{L})$ for some first-order language $\mathcal{L}$, then we may simply write $T_1 \sqsubseteq_\mathcal{L} T_2$ for $T_1 \sqsubseteq_{\mathcal{F}(\mathcal{L})} T_2$.*

Theorem 2.1.5 (Herbrand's theorem for existential formulas)**.** *Let $\varphi(\vec{x}, \vec{y})$ be a quantifier-free formula such that $\vdash (\exists \vec{y})\varphi(\vec{x}, \vec{y})$, then there is some $k \in \mathbb{N}$ and $\mathcal{L}(\varphi)$ terms $t_{i,j}(\vec{x})$ with $1 \leq i \leq n$ and $1 \leq j \leq k$ such that*

$$\vdash \bigvee_{j=1}^{k} \varphi(t_{1,j}, \dots t_{n,j}).$$

Proof. See for example [Sho67]. □

Let us now introduce two notions of completeness for theories.

Definition 2.1.6. *Let T be a theory, then we say that T is complete if $T \vdash \varphi$ or $T \vdash \neg\varphi$ for all $\mathcal{L}(T)$ sentences φ. Let Φ be a set of sentences, then T is complete for Φ if $T \vdash \varphi$ for all $\varphi \in \Phi$.*

Definition 2.1.7. *A set of sentences Φ is closed under modus ponens, if $\varphi, \varphi \to \psi \in \Phi$ implies $\psi \in \Phi$, for all formulas $\varphi, \varphi \to \psi$.*

In the following we will recall some model theoretic concepts and notations. Let $\mathcal{L}$ be a language, then an $\mathcal{L}$ structure is a pair $M = (D, I)$, where D is a non-empty set

and I is an interpretation function. The interpretation function I assigns to each symbol $S/n \in \mathcal{L}$ an interpretation S^I such that if S is predicate symbol, then $S^I \subseteq D^n$ and if S is a function symbol, then $S^I : D^n \to D$. We define $\mathcal{D}(M) = D$ and, moreover, for a symbol S of $\mathcal{L}$ we often write S^M instead of S^I. A variable assignment σ is a function from the variable symbols $\mathfrak{V}$ to $\mathcal{D}(M)$. We write $M, \sigma \models \varphi$ if φ if φ is true in M under the variable assignment σ. Let $\varphi(x_1, \ldots, x_n, \vec{y})$ be a formula and $d_1, \ldots, d_n \in D$, then we write $M, \{x_i \mapsto d_i \mid i = 1, \ldots, n\} \models \varphi$ (or $M \models \varphi(d_1, \ldots, d_n, \vec{y})$) if $M, \sigma \models \varphi$, for all variable assignments σ with $\sigma(x_i) = d_i$ for $i = 1, \ldots, n$. Thus, in particular, $M \models \varphi$ if $M, \sigma \models \varphi$, for all variable assignments σ. In other words free variables are treated as universally quantified. Let $t(x_1, \ldots, x_n)$ be a term and $d_1, \ldots, d_n$ a finite sequence of elements of $\mathcal{D}(M)$, then we write $t^M(\vec{d})$ to denote the element $b \in \mathcal{D}(M)$ such that $M, \{x_i \mapsto d_i \mid i = 1, \ldots, n\} \models t = b$.

Let Φ be a set of $\mathcal{L}$ formulas, then we write $M \models \Phi$ if $M \models \varphi$, for all $\varphi \in \Phi$. Moreover, let Ψ be a set of formulas, then we write $\Psi \models \Phi$ if for every model M of Ψ we have $M \models \Phi$.

Definition 2.1.8. *Let $\mathcal{L}, \mathcal{L}'$ be languages with $\mathcal{L}' \supseteq \mathcal{L}$. A $\mathcal{L}'$ structure M is an expansion of a $\mathcal{L}$ structure N if $\mathcal{D}(M) = \mathcal{D}(N)$ and $S^M = S^N$ for each symbol S of $\mathcal{L}$. Moreover, we define $M|\mathcal{L} = N$.*

Definition 2.1.9. *Let $\mathcal{L}$ be a language and M an $\mathcal{L}$ structure, then we define*

$$\mathrm{Th}(M) \coloneqq \{\varphi \mid M \models \varphi, \varphi \text{ is a sentence}\}.$$

2.2 Skolemization

In this thesis we essentially use inner Skolemization with canonical names. On the one hand this form of Skolemization is convenient from a theoretical point of view, because it can be described as a function on formulas. In particular, the canonical naming schema for Skolem symbols allows us to be precise about the languages generated during the saturation processes considered in this article. On the other hand, inner Skolemization performs comparatively well with respect to proof complexity [BL94], and furthermore using canonical Skolem symbols does not increase proof complexity. Hence, this form of Skolemization is also a reasonable choice from the perspective of automated deduction. This form of Skolemization that we use in this thesis is quite similar to the ϵ-calculus (see [MZ06]).

We start by defining an operator describing all the Skolem symbols that can be obtained by Skolemizing a single quantifier over a given language $\mathcal{L}$. This operator is then iterated on the language $\mathcal{L}$ in order to produce all the Skolem symbols that are required to Skolemize $\mathcal{L}$ formulas.

Definition 2.2.1. *Let $\mathcal{L}$ be a first-order language, then we define*

$$\mathfrak{S}_Q(\mathcal{L}) \coloneqq \{\mathfrak{s}_{(Qx)\varphi}/n \mid \varphi \text{ is an } \mathcal{L} \text{ formula}, |\mathrm{FV}((Qx)\varphi)| = n\},$$

where $Q \in \{\forall, \exists\}$. *We set* $\mathfrak{S}(\mathcal{L}) \coloneqq \mathfrak{S}_\forall(\mathcal{L}) \cup \mathfrak{S}_\exists(\mathcal{L})$. *Now we define* $sk(\mathcal{L}) \coloneqq \mathcal{L} \cup \mathfrak{S}(\mathcal{L})$. *By* $sk^i(\mathcal{L})$ *we denote the* i*-fold iteration of the* sk *operation. Finally, we define* $sk^\omega(\mathcal{L}) \coloneqq \bigcup_{i<\omega} sk^i(\mathcal{L})$. *We call the stage of a symbol the least* $i \in \mathbb{N}$ *such that the symbol belongs to the language* $sk^i(\mathcal{L})$. *A first-order language* $\mathcal{L}$ *is Skolem-free if it does not contain any of its Skolem symbols, that is, if* $\mathcal{L} \cap \mathfrak{S}(sk^\omega(\mathcal{L})) = \varnothing$.

Now we can define the universal and existential Skolem form of a formula.

Definition 2.2.2. *We define the functions* $sk^\forall, sk^\exists : \mathcal{F}(sk^\omega(\mathcal{L})) \to \mathcal{F}(sk^\omega(\mathcal{L}))$ *mutually inductively as follows*

$$\begin{aligned}
sk^Q(P(\vec{t})) &\coloneqq P(\vec{t}),\\
sk^Q(A \wedge B) &\coloneqq sk^Q(A) \wedge sk^Q(B),\\
sk^Q(A \vee B) &\coloneqq sk^Q(A) \vee sk^Q(B),\\
sk^Q(\neg A) &\coloneqq \neg sk^{\overline{Q}}(A),\\
sk^Q((Qx)A(x,\vec{y})) &\coloneqq sk^Q(A(\mathfrak{s}_{(Qx)A(x,\vec{y})}(\vec{y}),\vec{y})), \qquad (*)\\
sk^Q((\overline{Q}x)A) &\coloneqq (\overline{Q}x)sk^Q(A),
\end{aligned}$$

for $Q \in \{\forall, \exists\}$, $\overline{\forall} = \exists$, $\overline{\exists} = \forall$, *and where in* $(*)$ $\vec{y}$ *are* exactly *the free variables of* $(Qx)A$. *Let* Φ *be a set of formulas, then we define* $sk^Q(\Phi) \coloneqq \{sk^Q(\varphi) \mid \varphi \in \Phi\}$.

Before we discuss some details of the $sk^\exists$ function in more detail, we will look at an example that illustrates how the function $sk^\exists$ operates.

Example 2.2.3. *Let* $P/3$ *be a predicate symbol, then the existential Skolem form of the sentence* $(\exists x)(\forall y)(\exists z)P(x,y,z)$ *is given by*

$$\begin{aligned}
sk^\exists((\exists x)(\forall y)(\exists z)P(x,y,z)) &= sk^\exists((\forall y)(\exists z)P(c,y,z))\\
&= (\forall y)(sk^\exists((\exists z)P(c,y,z))))\\
&= (\forall y)(sk^\exists(P(c,y,f(y)))) = (\forall y)P(c,y,f(y)).
\end{aligned}$$

where $c = \mathfrak{s}_{(\exists x)(\forall y)(\exists z)P(x,y,z)}$ *and* $f = \mathfrak{s}_{(\exists z)P(c,y,z)} = \mathfrak{s}_{(\exists z)P(\mathfrak{s}_{(\exists x)(\forall y)(\exists z)P(x,y,z)},y,z)}$.

Observe that the symbols that are introduced by $sk^\exists$ depend on the names of the variables. Thus, in particular, the symbols introduced for two formulas that only differ in the names of bound variables may not be the same. For example, let P be a unary predicate symbol, then

$$sk^\exists((\exists x)P(x)) = P(\mathfrak{s}_{(\exists x)P(x)}) \neq P(\mathfrak{s}_{(\exists y)P(y)}) = sk^\exists((\exists y)P(y)).$$

Clearly, we could build the equivalence of formulas up to renaming into the Skolemization function. However, we prefer not to draw logical reasoning into the definition of the Skolemization function. Identification of provably equivalent formulas can be added by means of additional axioms, such as the Skolem axioms given in Definition 2.2.7.

Let us now recall some basic properties of Skolemization.

Definition 2.2.4. *Let φ be a formula, then a subformula occurrence of the form $(\forall x)\psi$ in φ is called a strong quantifier, if it occurs with positive polarity, otherwise it is called a weak quantifier. Dually, a subformula occurrence of the form $(\exists x)\psi$ in φ is called a weak quantifier if it occurs with positive polarity, otherwise it is called a strong quantifier.*

Lemma 2.2.5. *Let φ be a formula, then $sk^{\forall}(\varphi)$ does not contain strong quantifiers and $sk^{\exists}(\varphi)$ does not contain weak quantifiers.*

Proof. Straightforward. □

The following property of Skolemization is well-known.

Proposition 2.2.6. *Let $\mathcal{L}$ be first-order language and φ an $sk^{\omega}(\mathcal{L})$ formula. Then $\vdash sk^{\exists}(\varphi) \to \varphi$ and $\vdash \varphi \to sk^{\forall}(\varphi)$.*

In general we do not have the converse of the above implications. We will now introduce Skolem axioms.

Definition 2.2.7. *Let $\varphi(x, \vec{y})$ be a formula, then the Skolem axioms for φ are given by*

$$(\exists x)\varphi(x, \vec{y}) \to \varphi(\mathfrak{s}_{(\exists x)\varphi}(\vec{y}), \vec{y}), \qquad (S^{\exists}_x\varphi)$$
$$\varphi(\mathfrak{s}_{(\forall x)\varphi}(\vec{y}), \vec{y}) \to (\forall x)\varphi(x, \vec{y}). \qquad (S^{\forall}_x\varphi)$$

Let $\mathcal{L}$ be a language, then we define

$$\mathcal{L}\text{-SA} := \{(\forall \vec{y}) S^Q_x \varphi \mid Q \in \{\forall, \exists\}, \varphi(x, \vec{y}) \text{ is an } \mathcal{L} \text{ formula}\}.$$

The Skolem axioms allow us to also obtain the converse of Proposition 2.2.6.

Proposition 2.2.8. *Let $\mathcal{L}$ be a first-order language, φ an $sk^{\omega}(\mathcal{L})$ formula, and $Q \in \{\forall, \exists\}$. Then we have $sk^{\omega}(\mathcal{L})\text{-SA} \vdash \varphi \leftrightarrow sk^{Q}(\varphi)$.*

Proof. Straightforward. □

As shown by the following two lemmas the Skolem axioms essentially correspond to the existential Skolem form of the logical axioms $\varphi \to \varphi$.

Lemma 2.2.9. *Let φ be a formula and $Q \in \{\exists, \forall\}$, then $\vdash sk^{\exists}((Qx)\varphi \to (Qx)\varphi) \to S^Q_x\varphi$.*

Proof. We consider only the case for $Q = \forall$, the other case is dual. Let $\varphi(x, \vec{y})$ be a formula, assume $sk^{\exists}((\forall x)\varphi(x, \vec{y}) \to (\forall x)\varphi(x, \vec{y}))$ and $\varphi(\mathfrak{s}_{(\forall x)\varphi}(\vec{y}), \vec{y}))$. First of all, we have $sk^{\exists}((\forall x)\varphi \to (\forall x)\varphi) = sk^{\forall}(\varphi(\mathfrak{s}_{(\forall x)\varphi}(\vec{y}), \vec{y}))) \to sk^{\exists}((\forall x)\varphi)$. By Proposition 2.2.6 we have $\vdash sk^{\exists}((\forall x)\varphi) \to (\forall x)\varphi$, and hence we obtain $sk^{\forall}(\varphi(\mathfrak{s}_{(\forall x)\varphi}(\vec{y}), \vec{y}))) \to (\forall x)\varphi$. Again by Proposition 2.2.6 we have $\vdash \varphi(\mathfrak{s}_{(\forall x)\varphi}(\vec{y}), \vec{y}) \to sk^{\forall}(\varphi(\mathfrak{s}_{(\forall x)\varphi}(\vec{y}), \vec{y}))$. Thus, we obtain $(\forall x)\varphi$. □

Lemma 2.2.10. *Let $\mathcal{L}$ be a language, then*

$$sk^{\omega}(\mathcal{L})\text{-SA} \equiv \{(\forall \vec{z}) sk^{\exists}(\varphi \to \varphi) \mid \varphi(\vec{z}) \text{ is an } sk^{\omega}(\mathcal{L}) \text{ formula.}\}.$$

Proof. An immediate consequence of Proposition 2.2.8 and Lemma 2.2.9. □

Skolem axioms over a Skolem-free theory have the following well-known conservation property.

Proposition 2.2.11. *Let $\mathcal{L}$ be a Skolem-free first-order language and T be an $\mathcal{L}$ theory, then $sk^{\omega}(\mathcal{L})$-SA $+ T \equiv_{\mathcal{F}(\mathcal{L})} T$.*

Proof Sketch. Interpret the Skolem symbols as choice functions that select elements that satisfy their corresponding formulas. □

With the property above we now immediately obtain the well-known fact that Skolemizing a theory results in a conservative extension of that theory.

Lemma 2.2.12. *Let $\mathcal{L}$ be a Skolem-free language and T be an $\mathcal{L}$ theory, then*

$$sk^{\exists}(T) \equiv_{\mathcal{F}(\mathcal{L})} T.$$

Proof. The direction $sk^{\exists}(T) \sqsubseteq_{\mathcal{L}} T$ is an immediate consequence of Proposition 2.2.6. For the other direction we have $T \equiv^{\text{Prop. 2.2.11}}_{\mathcal{F}(\mathcal{L})} sk^{\omega}(\mathcal{L})\text{-SA}+T \equiv^{\text{Prop. 2.2.8}} sk^{\omega}(\mathcal{L})\text{-SA}+sk^{\exists}(T)$. Hence $T \equiv_{\mathcal{F}(\mathcal{L})} sk^{\exists}(T)$. □

This also immediately gives us the following weaker statement that is perhaps more familiar in automated deduction.

Corollary 2.2.13. *Let $\mathcal{L}$ be a Skolem-free language and T be theory, then T is consistent if and only if $sk^{\exists}(T)$ is consistent.*

2.3 A sequent calculus

In this section we introduce a sound and complete sequent calculus for first-order logic. Even though we introduce a concrete calculus for first-order logic we will whenever possible work with the more abstract provability relation given in Section 2.1. The calculus introduced in this section will be used occasionally to carry out arguments that rely on the structure of a proof.

Definition 2.3.1. *A sequent is an expression of the form $\Gamma \Rightarrow \Delta$, where Γ and Δ are finite multisets of formulas. The multiset Γ is called the antecedent and the multiset Δ is called the succedent.*

Informally, a sequent is to be interpreted as the implication of the disjunction of the formulas in the succedent by the conjunction of the formulas in the antecedent.

We will work with the following Gentzen system, which is essentially a variant of the calculus **G1c** given in [TS00] with atomic logical axioms extended by a cut rule and axioms for equality.

Definition 2.3.2. *The sequent calculus* **G** *consists of the following rules*
Axioms:

$$\frac{}{A \Rightarrow A}\ \mathrm{Ax} \qquad \frac{}{\bot \Rightarrow}\ \mathrm{L}\bot \qquad \frac{}{\Rightarrow \top}\ \mathrm{R}\top$$

$$\frac{}{\Rightarrow t = t}\ \mathrm{Refl} \qquad \frac{}{t = r, A[x/t] \Rightarrow A[x/r]}\ \mathrm{Eq}$$

Rules for weakening, contraction, and cut:

$$\frac{\Gamma \Rightarrow \Delta}{F, \Gamma \Rightarrow \Delta}\ \mathrm{LW} \qquad \frac{\Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, F}\ \mathrm{RW}$$

$$\frac{F, F, \Gamma \Rightarrow \Delta}{F, \Gamma \Rightarrow \Delta}\ \mathrm{LC} \qquad \frac{\Gamma \Rightarrow \Delta, F, F}{\Gamma \Rightarrow \Delta, F}\ \mathrm{RC}$$

$$\frac{\Gamma \Rightarrow \Delta, F \qquad F, \Lambda \Rightarrow \Pi}{\Gamma, \Lambda \Rightarrow \Delta, \Pi}\ \mathrm{Cut}$$

Rules for logical connectives and quantifiers:

$$\frac{F_i, \Gamma \Rightarrow \Delta}{F_0 \wedge F_1, \Gamma \Rightarrow \Delta}\ \mathrm{L}\wedge_i, i = 0, 1 \qquad \frac{\Gamma \Rightarrow \Delta, F \qquad \Gamma \Rightarrow \Delta, G}{\Gamma \Rightarrow \Delta, F \wedge G}\ \mathrm{R}\wedge$$

$$\frac{F, \Gamma \Rightarrow \Delta \qquad G, \Gamma \Rightarrow \Delta}{F \vee G, \Gamma \Rightarrow \Delta}\ \mathrm{L}\vee \qquad \frac{\Gamma \Rightarrow \Delta, F_i}{\Gamma \Rightarrow \Delta, F_0 \vee F_1}\ \mathrm{R}\vee_i,\ i = 0, 1$$

$$\frac{\Gamma \Rightarrow \Delta, F \qquad G, \Gamma \Rightarrow \Delta}{F \to G, \Gamma \Rightarrow \Delta}\ \mathrm{L}\to \qquad \frac{F, \Gamma \Rightarrow \Delta, G}{\Gamma \Rightarrow \Delta, F \to G}\ \mathrm{R}\to$$

$$\frac{F[x/t], \Gamma \Rightarrow \Delta}{\forall x F, \Gamma \Rightarrow \Delta}\ \mathrm{L}\forall \qquad \frac{\Gamma \Rightarrow \Delta, F[x/\alpha]}{\Gamma \Rightarrow \Delta, \forall x F}\ \mathrm{R}\forall$$

$$\frac{F[x/\alpha], \Gamma \Rightarrow \Delta}{\exists x F, \Gamma \Rightarrow \Delta}\ \mathrm{L}\exists \qquad \frac{\Gamma \Rightarrow \Delta, F[x/t]}{\Gamma \Rightarrow \Delta, \exists x F}\ \mathrm{R}\exists$$

where $\Gamma, \Delta, \Lambda, \Pi$ *stand for multisets of formulas,* F, G *stand for formulas,* A *stands for atomic formulas,* t, r *stand for terms, and for* $R \in \{\mathrm{L}\forall, \mathrm{R}\exists\}$ *the variable* α *is called the eigenvariable of* R *and* α *does not occur freely in the conclusion of* R.

We call the lower sequent of a rule its conclusion sequent. The proofs of **G** are defined inductively as usual: The axioms of **G** are proofs and the application of any of the rules of **G** to the conclusion sequents, of a suitable number of proofs, is a proof. For the sake of the presentation, we refer the reader to [TS00; TZ71] for a more detailed definitions of related proof-theoretic notions.

Let us now recall some basic properties of the sequent calculus **G**. We start with the soundness and completeness for first-order logic.

Lemma 2.3.3. *Let* φ *be a sentence, then* $\vdash \varphi$ *if and only if there exists a* **G** *proof of the sequent* $\Rightarrow \varphi$.

Proof. See for example [TS00]. □

Furthermore, the calculus **G** has the following form of cut elimination.

Definition 2.3.4. *In a cut inference the formula F is called the cut formula. We say that a* **G** *proof is in atomic cut-normal form (ACNF, for short) if all of its cut formulas are atomic.*

Lemma 2.3.5. *If a sequent $\Gamma \Rightarrow \Delta$ is provable in* **G***, then it has a* **G** *proof in ACNF.*

Proof. See [TS00]. □

We will now introduce some notions and results related to Skolemization.

Definition 2.3.6. *The inference rules* L$\exists$ *or* R$\forall$ *are called strong quantifier inference rules. Let π be a* **G** *proof, then by* $\mathrm{sqi}(\pi)$ *we denote the number of strong quantifier inferences in π.*

Lemma 2.3.7. *Let π be a* **G** *proof in ACNF of the sequent $\Pi, \Sigma \Rightarrow \Delta, \Lambda$, then there exists a proof π' in ACNF of $\Pi, sk^{\exists}(\Sigma) \Rightarrow sk^{\forall}(\Delta), \Lambda$ with $\mathrm{sqi}(\pi') \leq \mathrm{sqi}(\pi)$.*

Proof. We follow the ancestors of the formulas in Σ and Δ in π and replace eigenvariables of these ancestors by their respective Skolem terms. □

2.4 Saturation-based proof systems

Saturation is a technique of automated theorem proving that consists of computing the closure of a set of formulas under some inference rules. The saturation process goes on until some termination condition, such as the derivation of the empty clause, is met or until no more "new" formulas can be generated. Typically, saturation-based theorem provers operate in a clausal setting because clauses have a more uniform structure than formulas and are therefore better suited for automated proof search. In what follows we concentrate on the clausal refutational setting, because most state-of-the art theorem provers are refutation provers. That is, in order to determine for some theory T whether a given sentence φ is a consequence of T, the prover saturates the clause set $cnf(sk^{\exists}(T + \neg\varphi))$ until the empty clause is derived. However, our definitions can be easily adapted to the positive case by dualizing them, so as to cover for example connection-like methods. Saturation proof systems are usually based on a variant of the superposition calculus. In order not to get involved in the technical details of these saturation-based proof systems we will abstractly think of a such a prover as a state transition system whose current state is a set of derived clauses and whose state transitions are inference rules.

Definition 2.4.1 (Literals and clauses)**.** *Let L be a first-order language. An L literal is an L atom or the negation thereof. An L clause is a finite set of L literals. By $\Box$ we denote the empty clause, that is, $\bot$. An L clause set is a set of clauses. Let $\mathcal{C}$ be a clause set and D a clause, the we write $\mathcal{C} \vee D$ to denote the clause set $\{C \vee D \mid C \in \mathcal{C}\}$.*

Whenever the language L is clear from the context or irrelevant, we simply speak of clauses and clause sets instead of L clauses and L clause sets. Let P be an n-ary predicate symbol and $t_1, \ldots, t_n$ terms, then we define $\overline{P(t_1, \ldots, t_n)} = \neg P(t_1, \ldots, t_n)$ and $\overline{\neg P(t_1, \ldots, t_n)} = P(t_1, \ldots, t_n)$. A literal of the form $P(t_1, \ldots, t_n)$ (or $\neg P(t_1, \ldots, t_n)$) will be called a P-literal. If P is the equality symbol $=$, then we also call the literal an equational literal.

The following definition introduces some notation for writing clauses as an implication. This notation will be particularly useful in Chapter 5.

Definition 2.4.2. *Let C be a clause and $L_1, \ldots, L_n$ literals, then $C \leftarrow \bigwedge_{i=1}^n L_i$ denotes the clause $C \vee \bigvee_{i=1}^n \overline{L_i}$. Let $\mathcal{C}$ a clause set, then $\mathcal{C} \leftarrow \bigwedge_{i=1}^n L_i$ denotes the clause set $\{C \leftarrow \bigwedge_{i=1}^n L_i \mid C \in \mathcal{C}\}$.*

For a clause of the form $C \leftarrow \bigwedge_{i=1}^n L_i$ we call the formula $\bigwedge_{i=1}^n L_i$ the assertion part of the clause.

Definition 2.4.3. *By cnf we denote a fixed function that assigns to every sentence φ, that is free of weak quantifiers, clauses $C_1^\varphi(\vec{x}), \ldots, C_n^\varphi(\vec{x})$ such that*

$$\bigcup_{i=1}^n \mathcal{L}(C_i) \subseteq \mathcal{L}(\varphi) \text{ and } \vdash (\forall \vec{x}) \left(\bigwedge_{i=1}^n C_i^\varphi \right) \leftrightarrow \varphi.$$

Furthermore, let T be a $\forall_1$ theory, then $cnf(T) := \bigcup_{\varphi \in T} cnf(\varphi)$.

The function *cnf* fixed by the definition above could for example be the translation to conjunctive normal form that proceeds by moving negations inwards and by distributing disjunction over conjunction. We did not fix this particular translation because it is irrelevant for us how a conjunctive normal form is obtained as long as the translation preserves the language and is logically equivalent to the original sentence. In this thesis we ignore conjunctive normal form translations that do not preserve the language. The question how these more advanced transformations interact with induction is clearly very important and we hope to investigate it in the future.

Definition 2.4.4. *We fix a global linear order on clauses according to which we index clause sets. Let $\mathcal{C} = \{C_1(\vec{x}), \ldots, C_n(\vec{x})\}$ be a finite clause set, then we define $cnf^{-1}(\mathcal{C}) = (\forall \vec{x})(\bigwedge_{i=1}^n C_i)$.*

Lemma 2.4.5. *Let $\mathfrak{C}$ be a finite set of clause sets, then there exists a clause set $\mathcal{C}'$ such that $M \models \mathcal{C}'$ if and only if $M \models \mathcal{C}$, for some $\mathcal{C} \in \mathfrak{C}$.*

Proof. Let $\mathfrak{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_n\}$, now let $\psi := \bigvee_{i=1}^n cnf^{-1}(\mathcal{C}_i)$. Then ψ is logically equivalent to a $\forall_1$ formula ψ'. Now we define $\mathcal{C}' := cnf(\psi')$. It is clear that $M \models \mathcal{C}'$ if and only if $M \models \mathcal{C}_i$ for some $i \in \{1, \ldots, n\}$. □

We are now ready to introduce the notion of saturation systems that we will use in this thesis. In the literature we usually distinguish between inference rules and reduction rules but for our purposes this distinction is not necessary and permits us a more uniform presentation.

Definition 2.4.6. *A reduction rule R is a set of instances of the form*

$$\frac{C_1 \quad \dots \quad C_n}{\quad D_1, \dots, D_m},$$

where $C_1, \dots, C_n$ and $D_1, \dots, D_m$ are clauses.

A reduction rule of the form given in the definition above has the following intuitive meaning: If all the clauses $C_1, \dots, C_n$ are derived, then we may derive $D_1, \dots, D_m$ and we forget $C_1, \dots, C_n$. Inference rules are a special type of reduction rules that reintroduce their premises. In other words inference rules are reduction rules that do not forget any clauses. We denote inference rules analogously to reduction rules but we use a single horizontal bar instead of a double bar in order to indicate that the premises are not deleted. For example, we write

$$\frac{C_1 \quad \dots \quad C_n}{D_1, \dots, D_m},$$

with $C_1, \dots, C_n, D_1, \dots, D_m$ clauses, for the reduction rule

$$\frac{C_1 \quad \dots \quad C_n}{\quad C_1, \dots, C_n, D_1, \dots, D_m},$$

We will mostly work with inference rules whose instances are decidable in polynomial time.

Example 2.4.7. *The resolution rule* Res *is given by*

$$\frac{L_1 \vee C \qquad L_2 \vee D}{(C \vee D)\sigma}\ \text{Res},$$

where C, D are clauses, L_1, L_2 are literals and σ is a most general unifier of L_1 and $\overline{L_2}$.

Definition 2.4.8 (Saturation systems)**.** *A saturation system $\mathcal{S}$ is a set of reduction rules. Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two saturation-based proof systems, then by $\mathcal{S}_1 + \mathcal{S}_2$ we denote the system obtained by the union of the inference rules of $\mathcal{S}_1$ and $\mathcal{S}_2$.*

Definition 2.4.9 (Deduction, Refutation)**.** *Let $\mathcal{C}_0$ be a set of clauses and $\mathcal{S}$ a saturation-based proof system. A deduction from $\mathcal{C}_0$ in $\mathcal{S}$ is a finite sequence of clause sets $\mathcal{D}_0, \dots, \mathcal{D}_n$ such that $\mathcal{C}_0 = \mathcal{D}_0$ and for $0 \leq i < n$ there is an inference $C_1, \dots, C_n / D_1, \dots, D_m \in \mathcal{S}$ such that $C_1, \dots, C_n \in \mathcal{D}_i$ and $\mathcal{D}_{i+1} = \mathcal{D}_i \cup \{D_1, \dots, D_m\}$. We say that a clause C is derivable from $\mathcal{C}_0$ in $\mathcal{S}$ if there exists a deduction $\mathcal{D}_0, \dots, \mathcal{D}_n$ such that $C \in \mathcal{D}_n$. A deduction $\mathcal{D}_0, \dots, \mathcal{D}_n$ is called a refutation if $\square \in \mathcal{D}_n$.*

Let us finally introduce the notion of soundness for first-order logic and the notion of refutational completeness.

Definition 2.4.10. *Let $\mathcal{S}$ be a saturation system. We say that $\mathcal{S}$ is sound if whenever a clause C is derivable from a clause set $\mathcal{C}_0$ in $\mathcal{S}$, then $\mathcal{L}(C) \subseteq \mathcal{L}(\mathcal{C}_0)$ and $\mathcal{C}_0 \models C$. The saturation system $\mathcal{S}$ is said to be refutationally complete if there is a refutation from $\mathcal{C}_0$ if $\mathcal{C}_0$ is inconsistent.*

2.5 Induction

In this section we define the logical framework for induction that we will use in this thesis. In the context of automated inductive theorem proving one is typically interested in recursively defined datatypes such as natural numbers, lists, and trees. In this thesis we restrict ourselves to induction for natural numbers constructed by a nullary function symbol representing 0 and a unary function symbol s representing the successor operation for natural numbers. First of all, this avoids the notational complexity of general recursive datatypes and secondly, many interesting phenomena can already observed in this simpler setting. Nevertheless, the development of results for more complicated inductive types is certainly interesting but is left as future work (see Section 9.2).

Definition 2.5.1. *The language $\mathcal{L}_0$ consists of the nullary function symbol $0/0$ whose intended interpretation is the natural number 0 and the unary function symbol $s/1$ whose intended interpretation is the successor operation on natural numbers. Let f be a unary function symbol, $m \in \mathbb{N}$ and t a term, then $f^m(t)$ is defined inductively by $f^0(t) = t$ and $f^{i+1}(t) = f(f^i(t))$ for $i \in \mathbb{N}$. Finally, the expression $\overline{m}$ denotes the term $s^m(0)$.*

In Section 2.5.1 we define the inductive semantics that underlies the notion of induction considered in this thesis. In Section 2.5.2 we consider the effective forms of induction that we use in this thesis for the analysis of AITP systems.

2.5.1 Inductive semantics

In this section we will introduce the inductive semantics that underlies the notion of induction that we consider in this thesis. The inductive semantics that we define in this section serve only to set up a theoretical framework for induction in which we can, for example, speak about the soundness of AITP systems. In the remainder of the thesis we will however not work directly with these semantics. We first provide the notion of inductive structures and inductive consequences. After that, we show that these semantics correspond to the semantics induced by the second-order induction axiom. Finally, we recall some results concerning the computational properties of the inductive consequences.

In mathematical logic and in particular in arithmetic, the semantics are easily defined in terms of a unique standard model whose domain consists of the natural numbers. For inductive theorem proving the situation is slightly more complicated, because symbols do not necessarily have a unique interpretation over the natural numbers. To handle this technicality we first define the notion of inductive structures. These structures are essentially those first-order structures whose domain is constructed inductively from 0 and the successor operation s.

Example 2.5.2. *Consider for example the theory T whose axioms consist of the universal closure of the formulas (where A is a binary predicate symbol)*

$$A(0),$$
$$A(x) \to A(s(x)).$$

A natural inductive consequence is $(\forall x)A(x)$, hence for a function symbol f function symbol it is reasonable to expect that also $(\forall x)A(f(x))$ is an inductive consequence of T even though f does not occur in T.

We will in the following give several equivalent ways of defining inductive semantics and show that they are equivalent in some sense. This allows us to use the semantics that are most natural in a certain context.

We start with a definition that is based on fixed points and induces an operational construction of the intended interpretation of the inductive datatype. The semantics we define in this section are essentially a specialization of the ones given in [BS11]. For the sake of precision we repeat the construction, adapted to the setting of this thesis.

Definition 2.5.3 (Inductive structures)**.** *Let $\mathcal{L} \supseteq \mathcal{L}_0$ be a first-order language and M an $\mathcal{L}$ structure, then M is an inductive structure $\mathcal{L}$ if*

$$\mathcal{D}(M) = \{(s^n(0))^M \mid n \in \mathbb{N}\}.$$

The structure whose domain is $\mathbb{N}$ and that interprets 0 as the symbol 0 and s as the function $x \mapsto x + 1$ is an inductive structure. However, there are also other inductive structures. For example the structure M whose domain consists is $\{0, 1\}$ and interprets 0 as 0, $s^M(0)$ as 1, and $s^M(1)$ 1 is also an inductive structure. In AITP we are usually interested only in the structures whose domain are the natural numbers and interpret symbols in a certain way. We specify such properties by a, typically finite, background theory that provides, for example, the injectivity of s and defining equations for other function symbols.

We can now define the inductive consequences of a background theory.

Definition 2.5.4. *Let $\mathcal{L} \supseteq \mathcal{L}_0$ be a language, T an $\mathcal{L}$ theory, and φ a $\mathcal{L}$ sentence. We say that φ is an inductive consequence of T, in symbols $T \models_{\text{ind}} \varphi$ if for every inductive structure M, $M \models T$ implies $M \models \varphi$. By $[T]^{\mathcal{L}}_{\text{ind}}$ we denote the set of $\mathcal{L}$ sentences φ such that $T \models_{\text{ind}} \varphi$. If we are interested in the $\mathcal{L}(T)$ inductive consequences of T we also write $[T]_{\text{ind}}$ instead of $[T]^{\mathcal{L}(T)}_{\text{ind}}$.*

Example 2.5.5. *Consider again the theory T of Example 2.5.2. Let M be an inductive $\{0, s, f, A\}$ structure satisfying T. By a straightforward induction on i we show that $(s^i(0))^M \in A^M$ for all $i \in \mathbb{N}$. Thus, $A^M = \mathcal{D}(M)$, and therefore $M \models (\forall x)A(x)$. Hence, $T \models_{\text{ind}} A(f(x))$.*

Let us now discuss another common definition of inductive semantics in terms of the second-order induction axiom. This definition of inductive semantics is more natural for the logical formalism developed in this thesis.

Definition 2.5.6. *The second-order induction axiom* $\mathrm{IND}_{\mathbf{2}}$ *is the second-order sentence*

$$(\forall X)((X(0) \wedge (\forall x)(X(x) \to X(s(x)))) \to (\forall x)X(x)),$$

where the variable $X/1$ *is a variable that ranges over sets.*

Second-order semantics are defined as usual, in the sense that set variables quantify over subsets of the domain of a structure. It is straightforward to see that the semantics induced by the second-order induction axiom are equivalent to the fixed-point semantics defined above.

Lemma 2.5.7. *Let* $\mathcal{L} \supseteq \mathcal{L}_0$ *be language and* M *an* $\mathcal{L}$ *structure, then* M *is an inductive structure if and only if* $M \models \mathrm{IND}_{\mathbf{2}}$.

Proof. We first show the "only if" direction. Assume that M is an inductive structure. Let $X \subseteq \mathcal{D}(M)$ such that $0^M \in X$ and for every $d \in \mathcal{D}(M)$ we have $d \in X$ implies $s^M(d) \in X$. It suffices to show that $\mathcal{D}(M) \subseteq X$ By a straightforward induction on i we show that $(s^i(0))^M \in X$ for all $i \in \mathbb{N}$. Thus $\mathcal{D}(M) = X$. For the "if" direction it suffices to observe $X = \{(s^n(0))^M \mid n \in \mathbb{N}\}$ contains 0^M and is closed under the function s^M. Hence, by $\mathrm{IND}_{\mathbf{2}}$ we have $\mathcal{D}(M) = X$. Therefore M is an inductive structure. □

The lemma above gives us another way of writing the inductive entailment.

Proposition 2.5.8. *Let* T *be a theory and* φ *a sentence, then* $T \models_{\mathrm{ind}} \varphi$ *if and only if* $T + \mathrm{IND}_{\mathbf{2}} \models \varphi$.

Proof. An immediate consequence of Lemma 2.5.7. □

With these concepts we can now formulate the task addressed by inductive theorem proving more precisely. A problem for an inductive theorem prover consists of a finite set of axioms T—the background theory—and a sentence φ to be proved. An inductive theorem prover attempts to determine whether $T \models_{\mathrm{ind}} \varphi$, that is, $\varphi \in [T]_{\mathrm{ind}}^{\mathcal{L}(T) \cup \mathcal{L}(\varphi)}$. It is well-know that by Gödel's incompleteness theorem [Göd31] this problem is in general not semi-decidable. Moreover, by Tarski's undefinability theorem [Tar36] the inductive consequences of a background theory are in general not even arithmetically definable.

2.5.2 Induction schema and rules

In this section we will define the effective systems for induction that we use throughout the thesis. We introduce only the most important variants here. Other variants will be introduced when needed.

The most well-known form of effective induction is the first-order induction schema. This induction schema is essentially axiomatized by first-order instances of the second-order induction axiom. In particular, we parameterize the first-order induction schema by a set of induction formulas. This allows us to consider various induction schemes by varying the set of induction formulas.

Definition 2.5.9. *Let $\varphi(x, \vec{z})$ be a formula, then the formula $I_x\varphi$ is given by*

$$(\varphi(0, \vec{z}) \wedge \forall x(\varphi(x, \vec{z}) \to \varphi(s(x), \vec{z}))) \to \forall x\varphi(x, \vec{z}).$$

In the above definition we call φ the induction formula, x the induction variable, and $\vec{z}$ the induction parameters. Let Φ be a set of formulas, then the theory Φ-IND is axiomatized by the universal closure of the formulas $I_x\varphi$ for $\varphi(x, \vec{z}) \in \Phi$ and the parameter-free induction schema Φ-IND$^-$ is axiomatized by the sentences $I_x\varphi$ for $\varphi(x) \in \Phi$.

Depending on the choice of the set of induction formulas Φ the theories Φ-IND and Φ-IND$^-$ are not necessarily equivalent. For example, when we are dealing with induction formulas of restricted quantifier complexity such as $\exists_k(L)$-IND, then its parameter-free counterpart $\exists_k(L)$-IND$^-$ may be a weaker theory [KPD88]. Parameter-free induction schemata have been investigated in mathematical logic [Ada87; KPD88; Bek97b; Bek99; CFM11].

Another flavor of induction, that we will work with in this thesis, is induction based on a Hilbert-style inference rule. The idea of Hilbert-style inference rules and in particular of induction rules is made explicit in the following two definitions.

Definition 2.5.10. *A (Hilbert-style) inference rule R is a set of tuples of the form Γ/γ_0 called the instances of R, where $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$ is a finite set of sentences and γ_0 is a sentence. Let T be a theory, then the theory of unnested applications $[T, R]$ of the inference rule R over the theory T is axiomatized by*

$$T + \{\varphi \mid T \vdash \Gamma, \Gamma/\varphi \in R\}.$$

Let $[T, R]_0 := T$ and $[T, R]_{n+1} = [[T, R]_n, R]$, then we define $T + R := \bigcup_{n\geq 0}[T, R]_n$.

Let R be an inference rule and $\Gamma/\gamma_0 \in R$, then the intended meaning of the rule instance Γ/γ_0 is that whenever all the sentences in Γ are derived, then we can derive γ_0. The instance Γ/γ_0 will also be written as

$$\frac{\gamma_1 \quad \cdots \quad \gamma_n}{\gamma_0}$$

Definition 2.5.11. *Let Φ be a set of formulas, then the rule Φ-IND^R consists of the instances of the form*

$$\frac{\forall\vec{z}\gamma(0, \vec{z}) \qquad \forall\vec{z}\forall x(\gamma(x, \vec{z}) \to \gamma(s(x), \vec{z}))}{\forall\vec{z}\forall x\gamma(x, \vec{z})},$$

with $\gamma \in \Phi$ and where the variable x is called the induction variable and the variables $\vec{z}$ are called the induction parameters. The corresponding parameter-free rule Φ-IND^{R-} is defined by requiring that the formula φ has no free variables besides the induction variable.

Depending on the choice of the background theory and the set of induction formulas Hilbert-style induction rules are in general not as strong as the corresponding induction

schema, see [Par72]. Induction rules have been studied in mathematical logic, see for example [Sho58; She63; Par72; Bek97a; Jeř20].

A notion of inductivity given in the following definition, essentially corresponds to the applicability of an instance of the induction rule.

Definition 2.5.12. *Let T be a theory. A formula $\varphi(x, \vec{z})$ is T-inductive in x if $T \vdash \varphi(0, \vec{z})$ and $T \vdash \varphi(x, \vec{z}) \to \varphi(s(x), \vec{z})$. Whenever the induction variable x is clear from the context we simply say that φ is inductive in T.*

2.6 Arithmetic

In this section we introduce some notions related to theories of arithmetic that we use throughout the thesis. We introduce the setting of linear arithmetic which will serve to provide examples and unprovability results.

Usually we work with theories that extend the following theory which ensures that zero does not have a successor and that distinct elements have distinct successors.

Definition 2.6.1. *The theory $\mathcal{A}_0$ is axiomatized by the universal closure of the following formulas*

$$0 \neq s(x), \tag{A0.1}$$
$$s(x) = s(y) \to x = y. \tag{A0.2}$$

Definition 2.6.2. *The sentence* Pred *is* $(\forall x)[x = 0 \vee (\exists y)x = s(y)]$ *and the set* ACY *consists of the sentences of the form* $(\forall x)x \neq s^{k+1}(x)$ *with* $k \in \mathbb{N}$.

An interesting result obtained by Herbrand in 1930 is the following.

Theorem 2.6.3. *The theory* $\mathcal{A}_0 + \text{Pred} + \text{ACY}$ *is complete.*

Proof. See [Her30]. □

Corollary 2.6.4. $\mathcal{A}_0 + \text{Pred} + \text{ACY} \equiv \mathcal{A}_0 + \mathcal{L}(\mathcal{A}_0)\text{-IND}$

Proof. An immediate consequence of Theorem 2.6.3. □

Let us now introduce the slightly more complex setting of linear arithmetic which extends the above setting by a binary function symbol for addition and a unary symbol for the predecessor operation. This setting has the advantage of being sufficiently complex to provide interesting properties while still having straightforward model theoretic properties.

Definition 2.6.5 (Language of linear arithmetic)**.** *The language $\mathcal{L}_{\text{LA}}$ of linear arithmetic is $\{0, s, p, +\}$, where $p/1$ is a function symbol whose intended interpretation is the predecessor function on natural numbers and $+/2$ is an infix function symbol whose intended interpretation is the addition of natural numbers. Let t be a term and $n \in \mathbb{N}$, then the*

term $n \cdot t$, representing the multiplication of t by the constant n is defined inductively by $0 \cdot t \coloneqq 0$ and $(i+1) \cdot t \coloneqq t + (i \cdot t)$.

Let us denote by $\mathbb{N}_{\mathcal{L}_{\mathrm{LA}}}$ the structure whose domain is the set of natural numbers and that interprets the symbols naturally. In particular, $\mathbb{N}_{\mathcal{L}_{\mathrm{LA}}}$ interprets the symbol p as the function $x \mapsto x \dot{-} 1$, where $\dot{-}$ denotes cut-off subtraction, that is,

$$x \dot{-} y = \begin{cases} 0 & \text{if } x \leq y \\ x - y & \text{otherwise} \end{cases}.$$

When it is clear from the context that we are working with the language $\mathcal{L}_{\mathrm{LA}}$, then we will denote $\mathbb{N}_{\mathcal{L}_{\mathrm{LA}}}$ by $\mathbb{N}$.

Definition 2.6.6. *An $\mathcal{L}_{\mathrm{LA}}$ theory T is sound if $\mathbb{N}_{\mathcal{L}_{\mathrm{LA}}} \models T$.*

The background theory of linear arithmetic that we will use in this thesis is given by the following definition.

Definition 2.6.7. *The $\mathcal{L}_{\mathrm{LA}}$ theory B is axiomatized by the universal closure of the formulas*

$$s(0) \neq 0, \tag{A1}$$
$$p(0) = 0, \tag{A2}$$
$$p(s(x)) = x, \tag{A3}$$
$$x + 0 = x, \tag{A4}$$
$$x + s(y) = s(x + y). \tag{A5}$$

In the following we will recall some basic properties of the theory B.

Lemma 2.6.8. *The theory B is sound.*

Proof. Straightforward. □

Lemma 2.6.9. *(i)* $\mathrm{A1} + \mathrm{A2} + \mathrm{A3} \vdash s(x) \neq 0$.

(ii) $\mathrm{A3} \vdash s(x) = s(y) \to x = y$.

Proof. For (i) assume that there exists x such that $s(x) = 0$, then by A2 and A3 we have $x = p(s(x)) = p(0) = 0$. Thus, $s(0) = 0$ which contradicts A1. For (ii) assume $s(x) = s(y)$, then we have $p(s(x)) = p(s(y))$ and by (A3) we obtain $x = y$. □

Lemma 2.6.10. *Let t be an $\mathcal{L}_{\mathrm{LA}}$ ground term, then there exists $k \in \mathbb{N}$ such that $B \vdash t = \overline{k}$.*

Proof. Proceed by induction on the structure of the term. □

Lemma 2.6.11. *The theory B is complete for $\mathrm{Th}(\mathbb{N}_{\mathcal{L}_{\mathrm{LA}}}) \cap \exists_1(\mathcal{L}_{\mathrm{LA}})$.*

Proof. Assume that $\mathbb{N} \models \exists \vec{x} \varphi(\vec{x})$, where φ is quantifier-free. Then there are $n_1, \ldots, n_k$ such that $\mathbb{N} \models \varphi(\overline{n}_1, \ldots, \overline{n_k})$. Hence, it suffices to show that $\mathbb{N}_{\mathcal{L}_{\mathrm{LA}}} \models \psi$ implies $B \vdash \psi$ for all quantifier-free sentences. We proceed by induction on the structure of the sentence ψ. The only interesting case is the case where ψ is an atom $t_1 = t_2$. By Lemma 2.6.10 there exist $k_1, k_2 \in \mathbb{N}$ such that $B \vdash t_1 = t_2 \leftrightarrow \overline{k_1} = \overline{k_2}$. If $k_1 = k_2$ we apply reflexivity. Otherwise, we apply Lemma 2.6.9.(ii) repeatedly and finally we use 2.6.9.(i). □

3 Induction, Saturation, and Skolemization

In this chapter we will study abstractly some phenomena about induction and Skolemization in saturation-based theorem provers. To our knowledge the interaction between induction and Skolemization has not been investigated in the related literature. Modern saturation theorem provers are highly complicated mechanisms. Integrating induction into such a prover is non-trivial, because of the many possible interactions with the internal mechanisms of such a prover, such as, for example, clause splitting, term ordering constraints, Skolemization, and definitional transformations. The results in this section allow us to consider some interactions in isolation of the complicated underlying mechanism. Moreover, the results obtained in this section will help us in Chapters 4 and 5 to guess upper bounds.

Induction can be integrated into a saturation proving system in different ways. One possibility is to contain the induction mechanism in a separate module that may use a saturation prover to discharge subgoals. Moreover, the induction module may receive additional information from the saturation prover, for instance information about failed proof attempts [Biu+86]. Another, currently more popular, way is to integrate the induction mechanism more tightly into the saturation system as some form of inference rule, that, upon some condition, selects some clauses out of the generated clauses and constructs an induction formula based on the selected clauses. After that, the resulting induction axiom is clausified and the clauses are added to the search space [KP13; Ker14], [RV19; Haj+20; Haj21; Haj+21b; HKV21], [Cru15], [Wan17], [EP20]. The systems differ in the heuristics that are used to construct the induction formula, in the shape of the resulting induction formulas and in the conditions upon which an induction is performed. For instance, Kersani and Peltier's method [KP13; Ker14] carries out an induction only once, namely when the generated clauses are sufficient to derive the empty clause. Thus this method does, technically speaking, not even generate clauses. Kersani and Peltier's method has other particularities that we will consider in Chapter 6.

In Section 3.1 we consider an induction rule that uses induction formulas over the language generated by the saturation prover and extends that language by generating clauses corresponding to an induction axiom. In particular we will consider whether the occurrences of Skolem symbols inside the induction formulas grant any additional power to the system. We leave the general case open but show that in many cases the system does not have any additional power. In Section 3.2 we consider an induction rule that restricts occurrences of Skolem symbols in induction formulas to ground terms but allows no induction parameters. We show that this restriction corresponds has the power of parameterized induction. The restriction of Skolem symbols to ground terms of

induction formulas is motivated by a similar restriction encountered in practical systems.

3.1 Unrestricted induction and Skolemization

In this section we will consider an induction rule that generates new clauses from induction axioms. The rule we consider in this section imposes almost no restrictions on the induction formulas, except that they belong to the language generated by the saturation system. In particular, this rule may allow Skolem symbols to occur in the induction formulas. We will characterize extensions of saturation systems by this induction rule in terms of a logical theory. This allows us to consider the conservativity of the system about which we give a partial but far-reaching answer.

Definition 3.1.1. *The induction rule* $\mathrm{IND_S}$ *for saturation systems is given by*

$$\frac{C_1 \quad \dots \quad C_n}{cnf(sk^{\exists}(I_x\varphi(x,\vec{z})))}\ \mathrm{IND_S}$$

where $C_1, \dots, C_n$ *are clauses and* $\varphi(x,\vec{z})$ *is a* $\mathcal{L}(C_1,\dots,C_n)$ *formula.*

The following example illustrates how to use the above induction rule.

Example 3.1.2. *Let us work in the setting of linear arithmetic and let* $\mathcal{S}$ *be a sound and refutationally complete saturation system. We will now outline a refutation in* $\mathcal{S}+\mathrm{IND_S}$ *of the clause set* $\mathcal{C}_0$ *given by*

$$cnf(sk^{\exists}(B+\neg(\forall x)(\forall y)x+y=y+x)).$$

Let $sk^{\exists}(\neg(\forall x)(\forall y)x+y=y+x)=(c_1+c_2\neq c_2+c_1)$*, then we have* $c_1\in L(\mathcal{C}_0)$ *and*

$$\mathcal{C}_0 \models c_1+c_2\neq c_2+c_1. \tag{3.1}$$

Let $\varphi_1(x) :\equiv (c_1+x=x+c_1)$*, then we may apply the induction rule* $\mathrm{IND_S}$ *to obtain the clause set* $\mathcal{C}_1 :\equiv \mathcal{C}_0\cup cnf(sk^{\exists}(I_x\varphi_1(x)))$*. Let* $sk^{\exists}(I_x\varphi_1(x))=\varphi_1(0)\wedge\varphi_1(c_3)\to\varphi_1(s(c_3))\to\forall x\varphi_1(x)$*, then we have* $c_3\in L(\mathcal{C}_1)$ *and furthermore by* (3.1) *we have*

$$\mathcal{C}_1 \models \neg\varphi_1(0)\vee\neg(\varphi_1(c_3)\to\varphi_1(s(c_3))). \tag{3.2}$$

Since $\mathcal{C}_1\models c_1=c_1+0$*, we have* $\mathcal{C}_1\models\varphi(c_1,0)\leftrightarrow c_1=0+c_1$*. Let* $\varphi_2(x) :\equiv x=0+x$*, the we apply the induction rule* $\mathrm{IND_S}$ *in order to obtain the clause set* $\mathcal{C}_2 :\equiv \mathcal{C}_1\cup cnf(sk^{\exists}(I_x\varphi_2))$*. Let* $sk^{\exists}(I_x\varphi_2) :\equiv \varphi_2(0)\wedge(\varphi_2(c_4)\to\varphi_2(s(c_4)))\to(\forall x)\varphi_2$*, then by* (3.2) *we have*

$$cnf(\mathcal{C}_2)\models\neg\psi(0)\vee\neg(\psi(c_4)\to\psi(s(c_4)))\vee\neg(\varphi_1(c_3)\to\varphi_1(s(c_3))). \tag{3.3}$$

Now observe that $B\models 0=0+0$ *and* $B\models 0+s(c_4)=s(0+c_4)$*. Hence,* $B\models c_4=0+c_4\to s(c_4)=s(0+c_4)$*, that is,* $B\models\varphi_2(c_4)\to\varphi_2(c_4)$ *and* $B\models\varphi_2(0)$*. Therefore, by*

(3.3) *we obtain*

$$\mathcal{C}_2 \models \neg(\varphi_1(c_3) \to \varphi_1(s(c_3))). \tag{3.4}$$

Recall that $\varphi_1(x) = (c_1 + x = x + c_1)$. *Since* $B \models c_1 + s(x) = s(c_3 + x)$, *we have by* (3.4), $\mathcal{C}_2 \models \varphi_1(c_3) \leftrightarrow s(c_3 + c_1) \neq s(c_3) + c_1$. *Let* $\varphi_3(x) = (s(c_3 + x) = s(c_3) + x)$, *then by the above we obtain*

$$\mathcal{C}_2 \models \neg\varphi_3(c_3). \tag{3.5}$$

Now we apply the induction rule IND_S *in order to obtain the clause set* $\mathcal{C}_3 := \mathcal{C}_2 \cup cnf(sk^\exists(I_x\varphi_3))$. *Let* $sk^\exists(I_x\varphi_3) = (\varphi_3(0) \wedge (\varphi_3(c_5) \to \varphi_3(c_5))) \to (\forall x)\varphi_3$, *then by* (3.5) *we have*

$$\mathcal{C}_3 \models \neg\varphi_3(0) \vee \neg(\varphi_3(c_5) \to \varphi_3(s(c_5))). \tag{3.6}$$

Since $cnf(B) \models s(c_3 + 0) = s(c_3) = s(c_3) + 0$, *we have* $\mathcal{C}_3 \models \varphi(0)$. *Moreover,* $cnf(B) \models s(c_3 + s(c_5)) = s(s(c_3 + c_5))$, *hence* $cnf(B) \models \varphi_3(c_5) \to \varphi_3(s(c_5))$. *Hence, by* (3.6), *we have* $\mathcal{C}_3 \models \bot$. *Hence, by the refutational completeness of* $\mathcal{S}$ *we obtain a refutation of* $\mathcal{C}_3$. *Therefore, by combining the applications of* IND_S *used to obtain* $\mathcal{C}_3$ *with the* $\mathcal{S}$ *refutation of* $\mathcal{C}_3$ *we obtain a* $\mathcal{S} + \mathrm{IND}_\mathrm{S}$ *refutation of* $\mathcal{C}_0$.

There are two important observation that we can make about this induction rule IND_S. First of all, in a saturation system with this induction rule, Skolemization may happen at any time and not just once before the saturation process begins as is the case in saturation systems for pure first-order logic. Secondly, the induction rule IND_S permits Skolem symbols to appear in induction formulas. In other words, the rule IND_S iteratively extends the language of the induction formulas by Skolem symbols. Interestingly, a similar situation has been considered in the literature on mathematical logic [Bek03]. In saturation systems for pure first-order logic, the role of Skolemization is clear: Given a sentence φ, Skolemization allows us to obtain an equiconsistent sentence $sk^\exists(\varphi)$ without existential quantifiers (see Corollary 2.2.13). In saturation systems with the induction rule IND_S the role of Skolemization is not so clear anymore. This raises the question how the extension of the language of induction formulas by Skolem symbols affects the power of the system. In Section 3.1.1 we will provide a characterization of refutation by saturation systems extended by IND_S in terms of a theory with induction. In Section 3.1.2 we compare the power of the rule IND_S with Skolem-free induction. Also note that this feature is not artificial but actually appears for example in [Cru17], and [RV19].

3.1.1 Logical characterization

A useful technique for analyzing AITP systems the reduction of the system to an "equivalent" logical theory. Alternatively, when such a theory cannot be found it is can still be useful to approximate the system by a logical theory as closely as possible. The construction of that theory usually reveals some essential features of a method. Moreover, we can then make use of powerful techniques from mathematical logic in order to study the theory. In this section we will provide proof of the following logical characterization for sound and refutationally complete saturation systems extended by the induction rule IND_S.

Theorem 3.1.3. *Let $\mathcal{S}$ be a sound and refutationally complete saturation system, T a theory and φ an $\mathcal{L}(T)$ sentence, then $\mathcal{S} + \mathrm{IND}_\mathrm{S}$ refutes $cnf(sk^\exists(T + \neg\varphi))$ if and only if $sk^\omega\,(\mathcal{L}(T) \cup \mathcal{L}_0)$-SA $+ T + sk^\omega\,(\mathcal{L}(T) \cup \mathcal{L}_0)$-IND $\vdash \varphi$.*

We start by providing a logical theory that simulates a sound saturation systems extended by the induction rule IND_S.

Lemma 3.1.4. *Let $\mathcal{S}$ be a sound saturation system and $\mathcal{C}_0, \ldots, \mathcal{C}_n$ a $\mathcal{S} + \mathrm{IND}_\mathrm{S}$ deduction, then $sk^\omega\,(L(\mathcal{C}_0) \cup \mathcal{L}_0)$-SA$+\mathcal{C}_0+sk^\omega\,(L(\mathcal{C}_0) \cup \mathcal{L}_0)$-IND $\vdash \mathcal{C}_n$ and $L(\mathcal{C}_n) \subseteq sk^\omega\,(\mathcal{L}(C_0) \cup \mathcal{L}_0)$.*

Proof. We proceed by induction on n. The base case is trivial. For the induction step we assume the claim for n and consider $\mathcal{C}_{n+1}$. If $\mathcal{C}_{n+1}$ is obtained by an inference from $\mathcal{S}$, then by the soundness of $\mathcal{S}$ we have $\mathcal{L}(\mathcal{C}_{n+1}) \subseteq \mathcal{L}(\mathcal{C}_n)$ and $\mathcal{C}_n \models \mathcal{C}_{n+1}$. Hence the claim follows directly from the induction hypothesis. Otherwise, if $\mathcal{C}_{n+1}$ is obtained by an application of IND_S, then we have $\mathcal{C}_{n+1} = \mathcal{C}_n \cup cnf(sk^\exists(I_x\varphi))$, where $\mathcal{L}(\varphi) \subseteq \mathcal{L}(\mathcal{C}_n) \subseteq sk^\omega\,(\mathcal{L}(\mathcal{C}_0) \cup \mathcal{L}_0)$. By the induction hypothesis we clearly have $\mathcal{L}(\mathcal{C}_{n+1}) \subseteq sk^\omega\,(\mathcal{L}(\mathcal{C}_0) \cup \mathcal{L}_0)$. By Proposition 2.2.8, we have $sk^\omega\,(L(\mathcal{C}_0) \cup \mathcal{L}_0)$-SA$+sk^\omega\,(L(\mathcal{C}_0) \cup \mathcal{L}_0)$-IND $\models cnf(sk^\exists(I_x\varphi))$. Hence, by the induction hypothesis

$$sk^\omega\,(L(\mathcal{C}_0) \cup \mathcal{L}_0)\text{-SA} + \mathcal{C}_0 + sk^\omega\,(L(\mathcal{C}_0) \cup \mathcal{L}_0)\text{-IND} \models \mathcal{C}_{n+1}. \qquad \square$$

Proposition 3.1.5. *Let $\mathcal{S}$ be a sound saturation system and T a theory. If $\mathcal{S} + \mathrm{IND}_\mathrm{S}$ refutes $cnf(sk^\exists(T))$, then the theory $sk^\omega\,(L(T) \cup \mathcal{L}_0)$-SA $+ T + sk^\omega\,(L(T) \cup \mathcal{L}_0)$-IND is inconsistent.*

Proof. Observe that $sk^\omega\left(\mathcal{L}(cnf(sk^\exists(T))) \cup \mathcal{L}_0\right) = sk^\omega\,(\mathcal{L}(T) \cup \mathcal{L}_0)$ and $sk^\omega\,(\mathcal{L}(T) \cup \mathcal{L}_0)$-SA$+$ $T \models cnf(sk^\exists(T))$. Now, the claim follows immediately from Lemma 3.1.4. $\square$

The converse of the proposition above is essentially an application of the compactness theorem followed by a construction of a refutation. However, we have to deal with some technicalities concerning the languages and the Skolem axioms. Namely, we need to ensure that the Skolem symbols appearing in induction formulas are introduced by the Skolemization of a suitable formula.

Lemma 3.1.6. *Let T be a theory such that $sk^\exists(T) + sk^\exists(sk^\omega\,(\mathcal{L}(T) \cup \mathcal{L}_0)$-IND$)$ is inconsistent, then there is a finite set of sentences $S_0 \subseteq sk^\exists(T)$ and formulas $\varphi_1(x, \vec{y})$, $\ldots$, $\varphi_n(x, \vec{y})$ such that $S_0 \cup \{sk^\exists((\forall\vec{y})I_x\varphi_i(x, \vec{y})) \mid 1 \leq i \leq n\}$ is inconsistent, and for $i = 1, \ldots, n$,*

$$\mathcal{L}(\varphi_i) \subseteq \mathcal{L}(S_0) \cup \{sk^\exists((\forall\vec{y})I_x\varphi_j) \mid j < i\} \subseteq sk^\omega\,(\mathcal{L}(T) \cup \mathcal{L}_0).$$

Proof. By the compactness theorem we obtain a finite set $S_0 \subseteq sk^\exists(T)$ and formulas $\varphi_1(x, \vec{y}), \ldots, \varphi_n(x, \vec{y})$ over the language $sk^\omega\,(\mathcal{L}(T) \cup \mathcal{L}_0)$ such that $S_0 \cup \{sk^\exists((\forall\vec{y})I_x\varphi_i)\}$ is inconsistent. Let $S_{i+1} = S_i \cup \{sk^\exists((\forall\vec{y})I_x\varphi_{i+1})\}$ for $i = 0, \ldots, n-1$. Now assume that $\mathcal{L}(\varphi_m) \not\subseteq \mathcal{L}(S_{m-1})$ for some $m \in \{1, \ldots, n\}$. We proceed by induction on the lexicographic order of pairs (k, l) where k is the maximal stage of the construction of the Skolem language of $\mathcal{L}(T) \cup \mathcal{L}_0$ (see Definition 2.2.1) of a symbol σ for which there exists

$m \in \{1, \dots, n\}$ such that $\sigma \in \mathcal{L}(\varphi_m)$ but $\sigma \notin \mathcal{L}(S_{m-1})$ and $l \geq 1$ is the number of such symbols. If $k = 0$, then there is $\sigma \in \mathcal{L}(T) \cup \mathcal{L}_0$ and such that $\sigma \in \varphi_m$ and $\sigma \notin S_m$ for some $m \in \{1, \dots, n\}$. If $\sigma \in \mathcal{L}(T)$, then there exists a sentence $\psi \in T$ with $\sigma \in \psi$. Now consider the sequence $S_0 \cup \{\psi\}, S_1, \dots, S_n$. If $l = 1$, then it has the measure (k', l') with $k' < k$. Otherwise, if $l > 1$, then it has the measure $(k, l-1)$. In both case we can apply the induction hypothesis. If $\sigma \in \mathcal{L}_0$, then let $S_1' \coloneqq S_0 \cup \{sk^{\exists}(I_x x = x)\}$ and consider the sequence $S_0, S_1', S_1, \dots, S_n$. If $l = 1$, then the sequence is a suitable sequence has the measure (k', l') for some $k' < k$ and $l' \geq 1$. Otherwise it has the measure $(k, l-1)$. In both cases we can apply the induction hypothesis to obtain the desired sequence. Now consider the case $k \geq 1$. There exists a stage $k-1$ formula $\psi(x, \vec{y})$ such that $\sigma = \mathfrak{s}_{(Qx)\psi}$ with $Q \in \{\forall, \exists\}$ and a least $m_0 \geq 0$ such that $\sigma \in \mathcal{L}(\varphi_{m_0})$ and $\sigma \notin \mathcal{L}(S_{m_0-1})$. Now we let

$$\varphi'_{m_0} \coloneqq \begin{cases} I_x(\forall \vec{y})(Qx)\psi, & \text{if } Q = \forall \\ I_x(\forall \vec{y})\neg(Qx)\psi, & \text{if } Q = \exists \end{cases}.$$

Moreover, we define $S'_{m_0} \coloneqq S_{m_0-1} \cup \{sk^{\exists}(I_x \varphi'_{m_0})\}$. In both cases we clearly have $\sigma \in \mathcal{L}(sk^{\exists}(\varphi'_{m_0}))$. Hence, the sequence $S_0, S_1, \dots, S_{m_0-1}, S'_{m_0}, S_{m_0}, S_{m_0+1}, \dots, S_n$, has the measure (k', l') with either $k' < k$ or $l' < l$. Thus we may apply the induction hypothesis in order to obtain a suitable sequence of finite sets of sentences. □

Our first observation is that induction axioms that do not bind a free variable of the inducted upon formula allow us to introduce all the Skolem symbols. The formulas of the form $sk^{\exists}(\varphi \to \varphi)$ are of interest because they correspond, roughly speaking, to Skolem axioms.

Lemma 3.1.7. *Let $\varphi(\vec{y})$ be a formula and x a variable which does not occur in φ. Then $\mathcal{L}(sk^{\exists}(I_x\varphi)) = \mathcal{L}(sk^{\exists}(\varphi \to \varphi))$ and moreover $\vdash sk^{\exists}(I_x\varphi) \leftrightarrow sk^{\exists}(\varphi \to \varphi)$.*

Proof. Since the variable x does not occur in φ, we clearly have

$$\begin{aligned} sk^{\exists}(I_x\varphi) &= sk^{\forall}(\varphi) \wedge sk^{\forall}(\forall x(\varphi \to \varphi)) \to sk^{\exists}(\forall x \varphi) \\ &= sk^{\forall}(\varphi) \wedge (sk^{\exists}(\varphi) \to sk^{\forall}(\varphi)) \to \forall x(sk^{\exists}(\varphi)). \end{aligned}$$

Since $sk^{\exists}(\varphi \to \varphi) = sk^{\forall}(\varphi) \to sk^{\exists}(\varphi)$ we clearly have $\mathcal{L}(sk^{\exists}(I_x\varphi)) = \mathcal{L}(sk^{\exists}(\varphi \to \varphi))$. Furthermore, $sk^{\exists}(I_x\varphi)$ clearly is logically equivalent to $sk^{\exists}(\varphi \to \varphi)$. □

Lemma 3.1.8. *Let $\mathcal{L}$ be a first-order language, then $sk^{\exists}(\mathcal{L}\text{-IND}) \vdash \mathcal{L}\text{-SA}$.*

Proof. Let $\varphi(x, \vec{y})$ be an $\mathcal{L}$ formula and let $Q \in \{\exists, \forall\}$. Work in the theory $sk^{\exists}(\mathcal{L}\text{-IND})$, then we have in particular $sk^{\exists}(I_z Qx\varphi)$, where z is a variable not occurring in φ. By Lemma 3.1.7 we obtain $sk^{\exists}(Qx\varphi \to Qx\varphi)$. Hence, $S^Q_x\varphi$ by Lemma 2.2.9. □

Proposition 3.1.9. *Let $\mathcal{S}$ be a refutationally complete saturation system and T a theory. If the theory $sk^{\omega}(\mathcal{L}(T) \cup \mathcal{L}_0)\text{-SA} + T + sk^{\omega}(\mathcal{L}(T) \cup \mathcal{L}_0)\text{-IND}$ is inconsistent, then $\mathcal{S}+\text{IND}_{\text{S}}$ refutes the clause set $cnf(sk(T))$.*

Proof. Assume that $sk^{\omega}\,(\mathcal{L}(T) \cup \mathcal{L}_0)$-SA+$T$+$sk^{\omega}\,(\mathcal{L}(T) \cup \mathcal{L}_0)$-IND is inconsistent. Then, by Lemma 3.1.8, the theory $sk^{\exists}(T) + sk^{\exists}(sk^{\omega}\,(\mathcal{L}(T) \cup \mathcal{L}_0)\text{-IND})$ is inconsistent. By Lemma 3.1.6 we obtain finite sets of formulas $S_0 \subseteq \cdots \subseteq S_n$ such that $S_0 \subseteq sk^{\exists}(T)$ and $S_{i+1} = S_i \cup \{sk^{\exists}(I_x\varphi_{i+1})\}$ with $\mathcal{L}(S_{i+1}) \subseteq sk^{\omega}\,(\mathcal{L}(T) \cup \mathcal{L}_0)$ and $\mathcal{L}(\varphi_{i+1}) \subseteq \mathcal{L}(S_i)$, for $i = 0, \ldots, n-1$. Let $\mathcal{C}_0 = cnf(S_0)$ and $C_{i+1} = cnf(sk^{\exists}(I_x\varphi_{i+1}))$ for $i = 0, \ldots, n-1$. Clearly, for $i \in \{0, \ldots, n-1\}$ we can obtain $\mathcal{C}_{i+1}$ from $\mathcal{C}_i$ by an application of IND_S. Moreover, $\mathcal{C}_n \models \bot$. Hence, $\mathcal{S}$ refutes $\mathcal{C}_n$. Thus we obtain a $\mathcal{S} + \text{IND}_\text{S}$ refutation of $cnf(sk^{\exists}(T))$. □

Remark 3.1.10. *In practice a system does not carry out induction on variables that do not occur in a formula. Instead some systems (for example [Cru15; Cru17]) provide a lemma rule that introduces the clauses of $cnf(sk^{\exists}(\varphi \to \varphi))$ into the search space.*

As an immediate consequence of the above propositions we obtain the following characterization of refutability in a sound and refutationally complete saturation based system extended by the induction rule IND_S.

Proof of Theorem 3.1.3. An immediate consequence of Proposition 3.1.5 and Proposition 3.1.9. □

As a corollary we obtain the soundness of the IND_S rule with respect to inductive semantics. This result is not surprising but provides a rigorous soundness proof for a large class of induction mechanisms. This result can then be used to obtain soundness for concrete methods by showing that a concrete induction rule can be simulated by the rule IND_S. Typically soundness arguments rely on some form of semantics but doing such a simulation gives us already all the upper bounds established for the simulating rule as a side effect. The simulation can take place in any sufficiently powerful but sound saturation system.

Corollary 3.1.11. *Let $\mathcal{S}$ be a sound saturation system, $\mathcal{L} \supseteq \mathcal{L}_0$ a language, T an $\mathcal{L}$ theory, and σ an $\mathcal{L}$ sentence. If $\mathcal{S} + \text{IND}_\text{S}$ refutes the clause set $cnf(sk^{\exists}(T + \neg\sigma))$, then $T \models_{\text{ind}} \sigma$.*

Proof. Let M be an $\mathcal{L}$ inductive model of T, that is, $M \models T + \text{IND}_\mathbf{2}$. We expand M to an $sk^{\omega}\,(\mathcal{L})$ structure M' by assigning to the Skolem symbols suitable Skolem functions just as in the model-theoretic proof of Proposition 2.2.11. Then $M' \models \mathcal{L}$-SA and moreover $M' \models sk^{\omega}\,(\mathcal{L})$-IND, since M has induction for all subsets of $\mathcal{D}\,(M)$. By Theorem 3.1.3 we then have $M' \models \sigma$. Since M' is an expansion of M, we have $M \models \sigma$. Thus, by Proposition 2.5.8 we are done. □

We conclude this section with a remark.

Remark 3.1.12. *In the presence of the Skolem axioms every formula is equivalent to an open formula. In particular, for a language $\mathcal{L}$, we have*

$$\mathcal{L}\text{-SA} + \text{Open}(sk^{\omega}(\mathcal{L}))\text{-IND} \vdash sk^{\omega}(\mathcal{L})\text{-IND}.$$

Thus, we can formulate Theorem 3.1.3 in a slightly more canonical way, by using $\mathrm{Open}(sk^\omega(\mathcal{L}))$-IND *in place of* $sk^\omega(\mathcal{L})$-IND. *In other words, in the presence of Skolem axioms Skolem symbols permit us to simulate quantification. Conceptually, we can thus split the unrestricted induction rule of Definition 3.1.1 into a lemma rule and an induction rule for clause sets.*

3.1.2 Conservativity

Now that we have characterized refutability in saturation systems extended by the rule $\mathrm{IND_S}$ we can consider whether the addition of Skolem symbols grants additional strength to the system. In other words, we will consider the following question.

Question 3.1.13. *Let* $\mathcal{L}$ *be a Skolem-free language and* T *an* $\mathcal{L}$ *theory, do we have*

$$\mathcal{L}\text{-SA} + T + sk^\omega(\mathcal{L})\text{-IND} \sqsubseteq_{\mathcal{L}} T + \mathcal{L}\text{-IND?}$$

In the following we give a partial answer to the above question, that covers many cases of practical relevance. The general case remains open. Our answer relies on the following idea: If a Skolem function is definable in terms of an $\mathcal{L}$ formula then wherever the Skolem symbol occurs we can instead use its definition and thus eliminate the symbol.

Definition 3.1.14. *Let* $\mathcal{L}$ *be a language, and* M *an* $\mathcal{L}$ *structure. A function* $f : M^k \to M$ *is called* $\mathcal{L}$*-definable in* M *if there exists an* $\mathcal{L}$ *formula* $\varphi(\vec{x}, y)$ *such that for all* $\vec{d} \in M^k$ *we have* $f(\vec{d}) = b$ *if and only if* $M \models \varphi(\vec{d}, b)$.

Definition 3.1.15. *Let* $\mathcal{L}$ *be a language. We say that a structure* M *has definable Skolem functions if for every* $\mathcal{L}$ *formula* $\varphi(\vec{x}, y)$ *there exists a function* $f : M^k \to M$ *that is* $\mathcal{L}$*-definable in* M *and*

$$M \models \exists y \varphi(\vec{d}, y) \to \varphi(\vec{d}, f(\vec{d})), \text{ for all } \vec{d} \in M^k.$$

In the following we will provide a proof for the proposition below. After that, we briefly discuss some relevant instances of this result.

Proposition 3.1.16. *Let* T *be a Skolem-free theory. If every model* M *of* $T + \mathcal{L}(T)$-IND *has definable Skolem functions, then*

$$(\mathcal{L}(T) \cup \mathcal{L}_0)\text{-SA} + T + sk^\omega(\mathcal{L}(T))\text{-IND} \equiv_{\mathcal{L}(T)} T + \mathcal{L}(T)\text{-IND}.$$

Let us start by showing that if a Skolem symbol is definable in a structure, then we can eliminate the symbol in the structure by replacing the symbol by its definition.

Lemma 3.1.17. *Let* M *be an* $\mathcal{L}$ *structure,* $\psi(\vec{x}, y)$ *be an* $\mathcal{L}$ *formula such that* $M \models \exists! y \psi(\vec{x}, y)$. *Let furthermore* f *be a function symbol and* $\varphi(\vec{z})$ *an* $\mathcal{L} \cup \{f\}$ *formula. Let* $M' \coloneqq M \cup \{f \mapsto f_\psi\}$, *where* $f_\psi(\vec{a}) = b$ *with* $b \in M$ *the only choice so that* $M \models \psi(\vec{a}, b)$. *Then there exists an* $\mathcal{L}$ *formula* $\theta(\vec{z})$ *such that* $M' \models \varphi(\vec{z}) \leftrightarrow \theta(\vec{z})$.

Proof. This is easily seen by first observing that $M' \models f(\vec{x}) = y \leftrightarrow \psi(\vec{x}, y)$. By [Hod97, pp. 51–52] there exists an unnested $\mathcal{L}$ formula $\varphi'(\vec{z})$ such that $\varphi' \leftrightarrow \varphi$. In particular, the

symbol f occurs in φ' only in subformulas of the form $f(\vec{x}) = y$. Let the formula θ be obtained by replacing in φ' the subformulas of the form $f(\vec{x}) = y$ by $\psi(\vec{x}, y)$. Then θ is equivalent to φ in M' and does not contain f. □

In order to get rid of a symbol we need a defining formula. The assumption that a model has definable Skolem functions only provides definitions for Skolem symbols for $\mathcal{L}$ formulas. The definitions for other Skolem symbols that are introduced at later stages need to be constructed based on the definitions obtained for symbols of lower stages.

Lemma 3.1.18. *Let $\mathcal{L}$ be a Skolem-free first-order language and M an $\mathcal{L}$ structure with definable Skolem functions. Then there exists an expansion M' of M to $sk^\omega(\mathcal{L})$ such that $M' \models \mathcal{L}$-SA and for each Skolem symbol f of $sk^\omega(\mathcal{L})$, $f^{M'}$ is $\mathcal{L}$-definable in M'.*

Proof. We show by induction on $i \in \mathbb{N}$ that there is an expansion M_i of M to the language $sk^i(\mathcal{L})$ such that for each Skolem symbol f/m of $sk^i(\mathcal{L})$ there exists a formula $\psi_f(\vec{x}, y)$ such that $M_i \models f(x_1, \ldots, x_m) = y \leftrightarrow \psi_f(x_1, \ldots, x_m, y)$. The base case with $i = 0$ is trivial. For the induction step we assume the claim for i and consider the case for $i + 1$. Let $f \coloneqq \mathfrak{s}_{Qy\varphi(y,\vec{x})}$ be a Skolem symbol of $sk^{i+1}(\mathcal{L})$, that does not belong to $sk^i(\mathcal{L})$. Let $g_1/k_1, \ldots, g_n/k_n$ be the Skolem symbols occurring in the formula φ. Then clearly g_j belongs to $sk^i(\mathcal{L})$ for all $j = 1, \ldots, n$. By the induction hypothesis there exist $\mathcal{L}$ formulas $\psi_{g_j}(\vec{x}_j)$ such that $M_i \models g_j(x_1, \ldots, x_{k_j}) = y \leftrightarrow \psi_{g_i}(x_1, \ldots, x_{k_j}, y)$, for $j = 1, \ldots, n$. Then by repeated application of Lemma 3.1.17 to the formula φ, there exists an $\mathcal{L}$ formula $\psi_f(\vec{x}, y)$ such that $M_i \models \varphi(\vec{x}, y) \leftrightarrow \psi_f(\vec{x}, y)$. Since ψ_f is an $\mathcal{L}$ formula, M has definable Skolem functions, there exists a function $h : M^k \to M$ and an $\mathcal{L}$ formula $\delta_h(\vec{x}, y)$ such that h is defined in M by δ_h and $M \models \exists y \psi_f(\vec{x}, y) \to \psi_f(\vec{x}, h(\vec{x}))$. We set $f^{M_{i+1}} \coloneqq h$, then we have $M_{i+1} \models f(\vec{x}) = y \leftrightarrow \delta_h(\vec{x}, y)$. It remains to show that M_{i+1} satisfies the Skolem axiom for f. Suppose we have $M_i \models \exists y \varphi(\vec{d}, y)$, then we have $M_i \models \exists y \psi_f(\vec{d}, y)$. Hence $M_{i+1} \models \psi_f(\vec{d}, h(\vec{d}))$ and therefore $M_{i+1} \models \varphi(\vec{d}, f(\vec{d}))$. Hence $M_{i+1} \models \exists y \varphi(\vec{x}, y) \to \varphi(\vec{x}, f(\vec{x}))$. Finally, we obtain M' by $M' \coloneqq \bigcup_{i \geq 0} M_i$. □

Proving Proposition 3.1.16 is now just a matter of replacing Skolem symbols in induction formulas by their definitions.

Proof of Proposition 3.1.16. Let φ be an $\mathcal{L}$ formula such that $T + \mathcal{L}\text{-SA} + sk^\omega(\mathcal{L})\text{-IND} \vdash \varphi$. We proceed indirectly and assume $T + \mathcal{L}\text{-IND} \nvdash \varphi$. Then there exists a model M of $T + \mathcal{L}$-IND such that $M \not\models \varphi$. By Lemma 3.1.18 there exists an expansion M' of M to $sk^\omega(\mathcal{L})$ such that $M' \models \mathcal{L}$-SA and for every Skolem symbol f there exists an $\mathcal{L}$ formula $\delta_f(\vec{x}, y)$ such that $M' \models f(\vec{x}) = y \leftrightarrow \delta_f(\vec{x}, y)$. Let $\theta(x, \vec{z})$ be an $sk^\omega(\mathcal{L})$ formula and consider the induction axiom $I_x\theta(x, \vec{z})$. By Lemma 3.1.17 there exists an $\mathcal{L}$ formula $\theta'(x, \vec{z})$ such that $M' \models \theta(x, \vec{z}) \leftrightarrow \theta'(x, \vec{z})$. Hence we have $M' \models I_x\theta(x, \vec{z}) \leftrightarrow I_x\theta'(x, \vec{z})$. Since $M \models \mathcal{L}$-IND, we have $M' \models I_x\theta(x, \vec{z})$. Hence $M' \models sk^\omega(\mathcal{L})$-IND and therefore $M' \models T + \mathcal{L}\text{-SA} + sk^\omega(\mathcal{L})\text{-IND}$ but $M' \not\models \varphi$. Contradiction! □

In order to illustrate Proposition 3.1.16 we will consider some practically relevant special cases. An important special case of Proposition 3.1.16 is when the Skolem functions are definable already in a theory.

Definition 3.1.19. *Let T be a theory. We say that T has definable Skolem functions if for each $\mathcal{L}(T)$ formula $\varphi(\vec{x}, y)$, there exists an $\mathcal{L}(T)$ formula $\psi(\vec{x}, y)$ such that*

$$T \vdash \exists y \varphi(\vec{x}, y) \to \exists! y(\psi(\vec{x}, y) \wedge \varphi(\vec{x}, y)).$$

Proposition 3.1.20. *Let T be a Skolem-free theory such that $T \vdash (\exists! x)\theta(x)$ for some $\mathcal{L}(T)$ formula θ. If T has definable Skolem functions, then every model of T has definable Skolem functions.*

Proof. Let $\varphi(\vec{x}, y)$ be an $\mathcal{L}(T)$ formula, then there is $\psi(\vec{x}, y)$ such that $T \vdash (\exists y)\varphi(\vec{x}, y) \to (\exists! y)(\psi(\vec{x}, y) \wedge \varphi(\vec{x}, y))$. Now let $\psi'(\vec{x}, y) := (\neg(\exists y')\varphi(\vec{x}, y') \wedge \theta(y)) \vee ((\exists y')\varphi(\vec{x}, y') \wedge \psi(\vec{x}, y))$. Let us now show that $T \vdash (\exists! y)\psi'(\vec{x}, y)$. We work in T, if $(\exists y)\varphi(\vec{x}, y)$, then there is some y such that $\psi(\vec{x}, y)$ and $\varphi(\vec{x}, y)$. Hence we have $\psi'(\vec{x}, y)$. If there is no y such that $\varphi(\vec{x}, y)$, then we have $\psi'(\vec{x}, y_0)$ for some y_0 with $\theta(y_0)$. Assume that $\psi'(\vec{x}, y_1)$ and $\psi'(\vec{x}, y_2)$. If $(\exists y)\varphi(\vec{x}, y)$, then we have $\psi(\vec{x}, y_1)$ and $\psi(\vec{x}, y_2)$, thus $y_1 = y_2$. Otherwise if $\neg\exists y\varphi(\vec{x}, y)$, then we have $\theta(y_1)$, $\theta(y_2)$, thus $y_1 = y_2$. □

In particular, a theory has definable Skolem functions if it has a definable well-order. We simply need to define the Skolem functions in terms of the least of the candidate values in each point.

Definition 3.1.21. *Let $\mathcal{L}$ be a language, and $\theta(x, y)$ an $\mathcal{L}$ formula in two variables. For the sake of legibility we write $\theta(x, y)$ as $x \prec_\theta y$ and by $\forall x \prec_\theta y \psi(x, y)$ we abbreviate the formula $\forall x(x \prec_\theta y \to \psi(x, y))$. The total order axioms TO_θ for θ are given by*

$$x \not\prec_\theta x,$$
$$x \prec_\theta y \wedge y \prec_\theta z \to x \prec_\theta z,$$
$$x \prec_\theta y \vee y \prec_\theta x \vee x = y.$$

The least number principle $\mathcal{L}$-LNP_θ for $\theta(x, y)$ consists of the axioms

$$\forall \vec{z}(\exists x \psi(x, \vec{z}) \to \exists x(\psi(x, \vec{z}) \wedge \forall x' \prec_\theta x \neg\psi(x', \vec{z}))),$$

where ψ is an $\mathcal{L}$ formula. We define $\mathcal{L}$-$\mathrm{WO}_\theta := \mathrm{TO}_\theta + \mathcal{L}$-$\mathrm{LNP}_\theta$.

Proposition 3.1.22. *Let T be a Skolem-free theory. If there exists an $\mathcal{L}(T)$ formula $\theta(x, y)$ such that $T \vdash \mathcal{L}(T)$-WO_θ, then T has definable Skolem functions.*

Proof. Let $\varphi(\vec{x}, y)$ be an $\mathcal{L}(T)$ formula. We set $\psi(\vec{x}, y) = \varphi(x, y) \wedge \forall y' \prec_\theta y \neg\varphi(\vec{x}, y')$. Now work in T and assume that $\exists y\varphi(\vec{x}, y)$, then by the least number principle there exists y such that $\varphi(\vec{x}, y)$ and moreover $\forall y' \prec_\theta y \neg\varphi(\vec{x}, y')$. It remains to show that this y' is unique. Let u be an element with $\varphi(\vec{x}, y)$ and $\forall u' \prec_\theta u \neg\varphi(u, y)$. If $u \prec_\theta y$, then we obtain $\neg\varphi(\vec{x}, u)$. Analogously we obtain $\neg\varphi(\vec{x}, y)$ if $y \prec_\theta u$. Hence $u = y$. □

These results are quite far-reaching. For example, let T be a theory with $\mathcal{L}(T) \supseteq$

$\mathcal{L}_0 \cup \{+/2\}$ such that

$$T \vdash x + 0 = x,$$
$$T \vdash x + s(y) = s(x + y).$$

Then for $\theta := \exists z x + z = y$ we have

$$T + \mathcal{L}(T)\text{-IND} \vdash \mathcal{L}(T)\text{-WO}_\theta,$$

Therefore, extending the full induction principle by all the Skolem symbols based on such a theory results in a system that proves the same $\mathcal{L}(T)$ formulas as the Skolem-free system.

So far we have considered the effects of extending the full induction schema by all Skolem symbols. We have concluded that not only is this extension always sound but it is also conservative over the Skolem-free system in a setting where Skolem functions are definable in all models and in particular if the theory provides a well-order. We have left open the case where there are models in which a Skolem function is not definable.

3.2 Restricted induction and Skolemization

In the previous section we have considered some high-level questions about Skolemization in saturation theorem proving with an unrestricted induction rule. In this section we will consider a restriction of the rule IND_S that is based on the following design choices encountered in practical methods:

- Syntactical restriction of induction formulas: The methods presented in [RV19; Haj+20] restrict induction formulas to literals, [KP13; Ker14] restricts induction formulas to $\exists_1$ formulas, and [Cru15; Cru17] restricts induction formulas to $\forall_1$ formulas.

- Control over occurrences of Skolem symbols: The practical induction mechanisms exert control over occurrences of the induction Skolem symbols either by avoiding the introduction of Skolem symbols altogether [KP13; Ker14] or by introducing nullary Skolem symbols only [RV19; Haj+20], [Cru15; Cru17]. In particular none of these methods allows for parameters in the induction formula. As a consequence induction Skolem symbols trivially occur as subterms of ground terms.

Restrictions on the shape of the induction formulas is a feature that is common to all methods for automated inductive theorem proving because it is currently still difficult to search efficiently for a syntactically unrestricted induction formula. We incorporate this feature into the induction rule by parameterizing it by a set of formulas from which the induction formulas are constructed. The second feature is only slightly more complicated to generalize. If we are to allow induction formulas with quantifier alternations, then Skolemizing the corresponding induction axioms introduces non-nullary Skolem symbols. Hence, in subsequent inductions, variables may now occur in the scope of induction

Skolem symbols. Therefore, we generalize the second feature by explicitly requiring that variables do not occur within the scope of a Skolem symbol. In other words we require that Skolem symbols may appear in the induction formula only in subterms of ground terms. Both generalized features are captured by the following restricted induction rule.

Definition 3.2.1. *Let Φ be a set of formulas, then the ground induction rule Φ-GIND$_{\mathrm{S}}$ is given by*

$$\frac{C_1 \quad \dots \quad C_n}{cnf(sk^{\exists}(I_x\varphi(x,\vec{t})))}\ \Phi\text{-GIND}_{\mathrm{S}},$$

where $C_1, \dots, C_n$ are clauses, $\varphi(x,\vec{z}) \in \Phi$, and $\vec{t}$ is a vector of ground $\mathcal{L}(C_1,\dots,C_n)$ terms.

Remark 3.2.2. *This restriction on occurrences of Skolem symbols is not only motivated by abstracting the current practice in AITP, it is also of independent theoretical interest: As described in [Dow08], Skolemization without this restriction in simple type theory makes the axiom of choice provable, hence this restriction has been introduced in [Mil87]. This restriction is also used as an assumption for proving elementary deskolemization of proofs with cut in [BHW12; Kom22].*

Let us start with an example that illustrates the power of the rule Φ-GIND$_{\mathrm{S}}$. The induction rule Φ-GIND$_{\mathrm{S}}$ only generates parameter-free induction axioms, but on the other hand the generated induction axioms may contain Skolem symbols whose role is not yet clear at this point. Thus, we begin by comparing sound and refutationally complete saturation systems extended by the rule Φ-GIND$_{\mathrm{S}}$ with the induction schema Φ-IND^{-}. In the setting of linear arithmetic we readily obtain an example where both systems differ in strength.

Example 3.2.3. *Let $\mathcal{S}$ be a refutationally complete saturation system and let* $\mathrm{Can}_{\mathrm{r}}(x,y,z)$ *abbreviate the formula*

$$y + x = z + x \to y = z.$$

Now let us consider in the following the instance $\mathrm{Can}_{\mathrm{r}}(x,x,0)$ *given by*

$$x + x = 0 + x \to x = 0.$$

We will show that the saturation system $\mathcal{S} + \mathrm{Open}(\mathcal{L}(B))$-GIND$_{\mathrm{S}}$ refutes the clause set $cnf(sk^{\exists}(B + \neg\forall x \mathrm{Can}_{\mathrm{r}}(x,x,0)))$. Let $\mathcal{C}_0 \coloneqq cnf(sk^{\exists}(B + \neg\forall x \mathrm{Can}_{\mathrm{r}}(x,x,0))) = cnf(B + \neg\mathrm{Can}_{\mathrm{r}}(c_0,c_0,0))$. Now let

$$\varphi(x) \coloneqq c_0 + x = 0 + x \to c_0 = 0.$$

Then an application of $\mathrm{Open}(\mathcal{L}(B))$-GIND$_{\mathrm{S}}$ *yields the clause set $\mathcal{C}_1 \coloneqq \mathcal{C}_0 \cup cnf(sk^{\exists}(I_x\varphi))$. Let $sk^{\exists}(I_x\varphi) = (\varphi(0) \wedge (\varphi(c_1) \to \varphi(s(c_1)))) \to (\forall x)\varphi$. Since $\mathcal{C}_0 \models \mathrm{Can}_{\mathrm{r}}(c_0,c_0,0)$, we have*

$$\mathcal{C}_1 \models \neg\varphi(0) \vee \neg(\varphi(c_1) \to \varphi(s(c_1))).$$

Moreover, $\mathcal{C}_1 \models c_0 + 0 = c_0$ *and* $\mathcal{C}_1 \models 0 + 0 = 0$. *Hence,* $\mathcal{C}_1 \models \varphi(0)$ *and*

$$\mathcal{C}_1 \models \varphi(c_1) \wedge c_0 + s(c_1) = 0 + s(c_1) \wedge c_0 \neq 0.$$

Since $\mathcal{C}_1 \models s(c_0 + c_1) = c_0 + s(c_1) = 0 + s(c_1) = s(0 + c_1)$, *we have*

$$\mathcal{C}_1 \models c_0 + c_1 = 0 + c_1.$$

Thus, by $\mathcal{C}_1 \models \varphi(c_1)$ *we obtain* $\mathcal{C}_1 \models c_0 = 0$. *Therefore,* $\mathcal{C}_1 \models \bot$ *and by the refutational completeness of* $\mathcal{S}$ *we obtain an* $\mathcal{S}$ *refutation of* $\mathcal{C}_1$.

On the other hand we also have the following.

Lemma 3.2.4. $B + \mathrm{Open}(\mathcal{L}(B))\text{-IND}^- \not\vdash x + x = x + 0 \rightarrow x = 0$.

Proof. An immediate consequence of the more general Theorem 6.4.8 proved in Chapter 7. □

Remark 3.2.5. *The formula* $\mathrm{Can_r}(x, x, 0)$ *is interesting in the context of AITP because over the language* $\mathcal{L}(B)$ *(=* $\mathcal{L}_{\mathrm{LA}}$*) it requires induction on a more complex formula. We will later see in Chapter 6 that this formula is challenging for a family of AITP systems.*

Example 3.2.3 together with Lemma 3.2.4 show that a saturation system extended by the rule $\mathrm{Open}(\mathcal{L}(T))\text{-GIND}_{\mathrm{S}}$ can be more powerful than the theory $T + \mathrm{Open}(\mathcal{L}(T))\text{-IND}^-$. This suggests that Skolem symbols appearing in ground terms of the induction formulas have some of the power of induction parameters. In Section 3.2.1 we will confirm this intuition (see, in particular, Theorem 3.2.13).

3.2.1 Logical characterization

In this section we will provide a logical characterization for sound and refutationally saturation systems extended by the ground induction rule. This characterization in particular shows that Skolem symbols appearing in ground terms of induction formulas behave as induction parameters.

Let us again start by providing an upper bound for sound saturation systems extended by the ground induction rule.

Lemma 3.2.6. *Let* $\mathcal{S}$ *be a sound saturation-based proof system,* T *a theory, and* Φ *a set of formulas. If* $\mathcal{S} + \Phi\text{-GIND}_{\mathrm{S}}$ *refutes* $cnf(sk^{\exists}(T))$, *then the theory*

$$sk^{\omega}\left(\mathcal{L}(T) \cup \mathcal{L}(\Phi) \cup \mathcal{L}_0\right)\text{-SA} + T + \Phi\text{-IND}$$

is inconsistent.

Proof. Let $\mathcal{L}' \coloneqq sk^{\omega}\left(\mathcal{L}(T) \cup \mathcal{L}(\Phi) \cup \mathcal{L}_0\right)$, and $\mathcal{C}_0, \ldots, \mathcal{C}_n$ an $\mathcal{S} + \Phi\text{-GIND}_{\mathrm{S}}$ deduction from $cnf(sk^{\exists}(T))$. We show the stronger claim that $\mathcal{L}(\mathcal{C}_n) \subseteq \mathcal{L}'$ and $\mathcal{L}'\text{-SA} + T + \Phi\text{-IND} \models \mathcal{C}_n$. We proceed by induction on n. For the base case we have, by Proposition 2.2.8, $\mathcal{L}\text{-SA} + T \models \mathcal{C}_0$ and $\mathcal{L}(\mathcal{C}_0) \subseteq \mathcal{L}(sk^{\exists}(T)) \subseteq \mathcal{L}'$. For the induction step we assume

the claim for $\mathcal{C}_n$ and consider $\mathcal{C}_{n+1}$. If $\mathcal{C}_{n+1}$ is obtained by $\mathcal{S}$, then the claim follows trivially from the soundness of $\mathcal{S}$ and the induction hypothesis. Otherwise, if $\mathcal{C}_{n+1}$ is obtained by an application of Φ-GIND$_\mathrm{S}$, then $\mathcal{C}_{n+1} = \mathcal{C}_n \cup cnf(sk^{\exists}(I_x\varphi(x,\vec{t})))$, where $\varphi(x,\vec{y}) \in \Phi$ and $\vec{t}$ is a vector of ground $\mathcal{L}(\mathcal{C}_n)$ terms. Since, Φ-IND $\vdash (\forall\vec{y})I_x\varphi$, we have Φ-IND $\vdash I_x\varphi(x,\vec{t})$. By the induction hypothesis $\mathcal{L}(I_x\varphi(x,\vec{t})) \subseteq \mathcal{L}'$, hence, by Proposition 2.2.8, we obtain $\mathcal{L}'$-SA$+\Phi$-IND $\vdash sk^{\exists}(I_x\varphi(x,\vec{t}))$. Therefore, by the induction hypothesis we have $\mathcal{L}'$-SA $+ T + \Phi$-IND $\models \mathcal{C}_{n+1}$. □

In the proof of Lemma 3.2.6 the role of the Skolem axioms is peculiar in the sense that they allow us to obtain the Skolemization of an induction axiom after instantiating the induction parameters. However, the Skolem axioms introduced by the proof above seem to introduce more information about the Skolem symbols, than the Skolemization in the ground induction rule. This is in particular the case for the Skolem symbols appearing in T and Φ. Hence, it is an interesting question whether converse of the above lemma is true A positive answer would provide a better characterization than the one developed in this section.

Question 3.2.7. *Let $\mathcal{S}$ be a sound saturation-based proof system, T a theory, and Φ a set of formulas. Assume that the theory*

$$sk^{\omega}\,(\mathcal{L}(T) \cup \mathcal{L}(\Phi) \cup \mathcal{L}_0)\text{-SA} + T + \Phi\text{-IND}$$

is inconsistent. Does this imply that $\mathcal{S} + \Phi$-GIND$_\mathrm{S}$ refutes $cnf(sk^{\exists}(T))$? In other words do the Skolem axioms increase the refutational power?

In a practical setting the theory T is usually Skolem-free but the set of induction formulas Φ may contain Skolem symbols, for example those of the initial clause set. However, in many practical cases these Skolem symbols are absorbed by the induction schema in the sense that there is a Skolem-free set of formulas Φ' such that Φ-IND $\equiv$ Φ'-IND. An example of such a situation is when $\Phi = \mathrm{Open}(\mathcal{L}(sk^{\exists}(T)))$ and T is an $\exists_2$ theory, then Skolemizing T only generates nullary Skolem symbols that are absorbed by the induction parameters, that is, $\mathrm{Open}(\mathcal{L}(sk^{\exists}(T)))$-IND $\equiv \mathrm{Open}(\mathcal{L}(T))$-IND. In such a setting we obtain the following simpler upper bound.

Lemma 3.2.8. *Let $\mathcal{S}$ be a sound saturation system, T be a Skolem-free theory, Φ a set of formulas, and Φ' a Skolem-free set of formulas with Φ-IND $\equiv \Phi'$-IND. If $\mathcal{S} + \Phi$-GIND$_\mathrm{S}$ refutes the clause set $cnf(sk^{\exists}(T))$, then $T + \Phi$-IND is inconsistent.*

Proof. An immediate consequence of Lemma 3.2.6 and Proposition 2.2.11. □

In the following we will show, by a proof-theoretic argument, a dual form of the Lemma 3.2.8. In other words we will show that the Skolem symbols in ground terms behave as induction parameters. We introduce a partially prenexed form of the induction schema in which the strong quantifier of the induction step is pulled into the quantifier prefix. Moving this quantifier into the quantifier prefix will simplify the subsequent arguments.

Definition 3.2.9. *Let $\varphi(x, \vec{z})$ be a formula, then we define the formula $I'_x\varphi$ by*

$$I'_x\varphi \coloneqq (\exists x)(\underbrace{(\varphi(0,\vec{z}) \wedge (\varphi(x,\vec{z}) \to \varphi(s(x),\vec{z})))}_{J_x\varphi(x,\vec{z})} \to (\forall y)\varphi(y,\vec{z})).$$

Let Φ be a set of formulas, then the theory Φ-IND$'$ is axiomatized by the universal closure of the formulas $I'_x\varphi$ with $\varphi \in \Phi$.

This induction schema is clearly equivalent to the usual one given in Definition 2.5.9.

Lemma 3.2.10. Φ-IND $\equiv \Phi$-IND$'$.

Proof. Straightforward. □

Lemma 3.2.11. *Let Γ be a finite set of formulas not containing weak quantifiers such that Γ contains at least one nullary function symbol. Furthermore, let Φ a set of formulas and Λ be a finite subset of Φ-IND$'$ such that the sequent $\Gamma, \Lambda \Longrightarrow$ is provable. Then there exist formulas $\varphi_1(x, \vec{y})$, …, $\varphi_n(x, \vec{y}) \in \Phi$ and finite sequences of ground terms $\vec{t}_1$, …, $\vec{t}_n$ such that the sequent $\Gamma, sk^{\exists}(I_x\varphi_1(x, \vec{t}_1)), \ldots, sk^{\exists}(I_x\varphi_n(x, \vec{t}_n)) \Longrightarrow$ is provable and $\mathcal{L}(\vec{t}_i) \subseteq \mathcal{L}(\Gamma \cup \{sk^{\exists}(I_x\varphi_j(x, \vec{t}_j) \mid 1 \leq j < i\})$.*

Proof. Let μ be a proof in ACNF of the sequent $\Gamma, \Lambda \Longrightarrow$. We proceed by induction on the number of strong quantifier inferences of μ. For the base case assume that μ does not contain any strong quantifier inferences. Since Γ does not contain weak quantifiers, all the cut inferences in μ are atomic, and the sentences in Λ have a weak quantifier in their quantifier prefix, all the ancestors of Λ are introduced by weakening. Hence, by permuting the weak quantifier inferences, weakening inferences, and contraction inferences whose main formula is an ancestor of a formula in Λ toward the bottom of the proof we obtain a proof of the form

$$\dfrac{\dfrac{\dfrac{(\mu')}{\Gamma \Longrightarrow}}{\Gamma, \Lambda'' \Longrightarrow}\,\mathrm{L}\forall^*}{\Gamma, \Lambda' \Longrightarrow}\,\mathrm{LC}^* \quad ,$$

$$\dfrac{\dfrac{\dfrac{(\mu')}{\Gamma \Longrightarrow}}{\Gamma, \Lambda'' \Longrightarrow}\,\mathrm{LW}^*}{\Gamma, \Lambda' \Longrightarrow}\,\mathrm{L}\forall^* \quad ,$$

where in the part below μ' the formulas in Γ only appear in the context. Hence, the sequent $\Gamma \Longrightarrow$ is provable and we are done. Now suppose that μ contains a strong quantifier inference. Since μ is in ACNF we can permute the quantifier inferences downward in order to obtain a proof of the form

$$\dfrac{\dfrac{\dfrac{(\mu'(\alpha))}{\Gamma, J_x\varphi(\alpha,\vec{t}), \Lambda \Rightarrow}}{\Gamma, \exists x J_x\varphi(x,\vec{t}), \Lambda \Rightarrow}\,\mathrm{L}\exists}{\Gamma, (\forall\vec{z})(\exists x)J_x\varphi(x,\vec{z}), \Lambda \Rightarrow}\,\mathrm{L}\forall^*}{\Gamma, \Lambda \Rightarrow,}\,\mathrm{LC}$$

with $\mathrm{sqi}(\mu') < \mathrm{sqi}(\mu)$ and $\vec{t}$ a finite sequence of $\mathcal{L}(\Gamma \cup \Lambda)$ ground terms. Now let $c \coloneqq \mathfrak{s}_{(\forall x)(\varphi(x,\vec{t})\to\varphi(s(x),\vec{t}))}$, then let $\mu'' = \mu'(c)$ is a proof of $\Gamma, J_x\varphi(c,\vec{t}), \Lambda \Longrightarrow$ with $\mathrm{sqi}(\mu'') = \mathrm{sqi}(\mu')$. Since $sk^{\exists}((J_x\varphi)[\vec{y}/\vec{t}, \alpha/c])) = sk^{\exists}(I_x\varphi(x,\vec{t}))$, we obtain by Lemma 2.3.7 a proof ν of the sequent $\Gamma, sk^{\exists}(I_x\varphi(x,\vec{t})), \Lambda \Longrightarrow$ with $\mathrm{sqi}(\mu) \leq \mathrm{sqi}(\mu'') < \mathrm{sqi}(\mu)$. Now let $\sigma_1, \ldots, \sigma_k$ be all the symbols of $\mathcal{L}(\vec{t})$ such that $\sigma_i \notin \mathcal{L}(\Gamma)$ for $i = 1, \ldots, k$. Let $i \in \{1, \ldots, k\}$, if $\sigma_i \in \mathcal{L}_0$, then let $\psi_i(x_1, \ldots, x_m) \in \Phi$, otherwise let $\psi_i(x_1, \ldots, x_m) \in \Phi$ such that $\sigma_i \in \mathcal{L}(\psi_i)$. Now let the proof ν' be given by

$$\dfrac{\begin{array}{c}(\nu)\\ \Gamma, sk^{\exists}(I_x\varphi(x,\vec{t})), \Lambda \Rightarrow\end{array}}{\Gamma, sk^{\exists}(I_x\psi_1(d,\ldots,d)), \ldots, sk^{\exists}(I_x\psi_m(d,\ldots,d)), sk^{\exists}(I_x\varphi(x,\vec{t})), \Lambda \Rightarrow,}\ \mathrm{LW}^*$$

where d is a nullary function symbol of Γ. Clearly, $\mathrm{sqi}(\nu') < \mathrm{sqi}(\mu)$. Hence, we can apply the induction hypothesis to obtain the desired formulas. □

Proposition 3.2.12. *Let T be a theory such that $\mathcal{L}(sk^{\exists}(T))$ contains at least one constant symbol, Φ a set of formulas, and $\mathcal{S}$ a refutationally complete saturation system. If $T + \Phi\text{-IND}$ is inconsistent, then $\mathcal{S} + \Phi\text{-GIND}_{\mathrm{S}}$ refutes $cnf(sk^{\exists}(T))$.*

Proof. Assume $T + \Phi$-IND is inconsistent, then $sk^{\exists}(T) + \Phi\text{-IND}'$ is inconsistent as well. Hence, by the compactness theorem there exist finite sets of formulas $\Gamma' \subseteq sk^{\exists}(T)$ and $\Lambda \subseteq \Phi\text{-IND}'$ such that $\Gamma' \cup \Lambda$ is inconsistent. Now obtain Γ by adding to Γ' a sentence of $sk^{\exists}(T)$ containing a nullary function symbol. By the completeness of $\mathbf{G}$ the sequent $\Gamma, \Lambda \Longrightarrow$ is provable. We apply Lemma 3.2.11 in order to obtain formulas $\varphi_1(x,\vec{y})$, $\ldots$, $\varphi_n(x,\vec{y}) \in \Phi$ and finite sequences of terms $\vec{t}_1$, $\ldots$, $\vec{t}_n$ such that for $S_0 = \Gamma$ and $S_{i+1} = S_i \cup \{sk^{\exists}(I_x\varphi_{i+1}(x,\vec{t}_{i+1}))\}$, we have $\mathcal{L}(\vec{t}_{i+1}) \subseteq \mathcal{L}(S_i)$, for $0 \leq i < n$. Now we let $\mathcal{C}_0 = cnf(S_0)$ and infer $\mathcal{C}_i$ from $\mathcal{C}_{i-1}$ by an application of $\Phi\text{-GIND}_{\mathrm{S}}$, for $i = 1, \ldots, n$. Then $\mathcal{C}_n$ is inconsistent. Hence, by the refutational completeness of $\mathcal{S}$, we obtain an $\mathcal{S}$ refutation of $\mathcal{C}_n$. □

Let us summarize the results so far.

Theorem 3.2.13. *Let $\mathcal{S}$ be a sound and refutationally complete saturation system, T a Skolem-free theory such that $sk^{\exists}(T)$ contains a nullary function symbol, Φ a set of formulas, and Ψ a Skolem-free set of formulas such that $\Phi\text{-IND} \equiv \Psi\text{-IND}$, then $\mathcal{S} + \Phi\text{-GIND}_{\mathrm{S}}$ refutes $cnf(sk^{\exists}(T))$ if and only if $T + \Psi\text{-IND}$ is inconsistent.*

Proof. An immediate consequence of Lemma 3.2.8 and Proposition 3.2.12. □

The results above show that, in a refutational setting, allowing Skolem symbols to appear in ground terms of induction formulas corresponds exactly to induction with parameters. This confirms our initial intuition that Skolem symbols in ground terms behave like induction parameters.

Remark 3.2.14 (Delayed Skolemization)**.** *The Skolemization in the rule $\Phi\text{-GIND}_{\mathrm{S}}$ is interesting in the sense that it generates nullary Skolem symbols only, but requires Skolemization to take place after instantiation. The proof of Lemma 3.2.11 shows that this use*

of Skolemization essentially corresponds to the use of eigenvariables in the sense of the sequent calculus.

We conclude this section with a remark on a possible generalization of the results in this section. In the context of automated inductive theorem proving we are typically interested in results about induction with arbitrary constructors and not just natural numbers. Furthermore, one would, especially in Section 3.2, be interested in the various variants of induction schemata. It seems possible to generalize these results to any axiom schema. The only place where we have used a property of the induction schema is in Lemma 3.1.8, where the Skolemized induction schema is used to prove the Skolem axioms. Hence, we could introduce a lemma rule as discussed in Remark 3.1.10 to achieve the same effect.

4 Case study: Vampire

In the previous chapter we have studied two forms of induction rules occurring in saturation-based induction provers. In particular, we were able to give a Skolem-free characterization of sound and refutationally complete saturation systems extended by the induction rule Φ-GIND$_S$ when Φ is a set of Skolem-free formulas. In this section we will make use of this result in order to provide concrete unprovability results for some instances of the AITP systems described in [RV19; Haj+20; Haj+21b]. All of these methods are extensions of a saturation-based proof system by an induction rule that generates clauses corresponding to an induction axiom. These extensions are implemented as extensions of the first-order theorem prover Vampire [KV13].

A saturation prover may use mechanisms that extend the working language of the prover. For example, two common mechanisms are definitional translations of the input formulas and clause splitting mechanisms. As in the previous chapter we assume a setting where the clause normal form transformation extends the language by Skolem symbols only, which is consistent with the presentation in [RV19; Haj+20; Haj+21b]. In particular, the unprovability results developed in this chapter are based on concrete clause sets and therefore definitional translations do not apply. However, the setting used in this chapter ignores possible splitting mechanisms of the underlying calculus. In Chapter 5 we consider a similar system with a clause splitting mechanism that extends the language of the prover. In particular, it will be shown that the splitting mechanism does not affect overall power of the induction rule.

We start by considering in Section 4.1 the extension of sound saturation systems by the single-clause induction rules introduced in [RV19; Haj+20]. After that, we will consider in Section 4.2 variants of the more powerful rule introduced in [Haj+21b] that operates on multiple clauses simultaneously. Finally, we will provide in Section 4.3 a preliminary unprovability result for a variant of the single-clause induction rule that takes into account the underlying calculus and improves upon the unprovability results established in Section 4.1.

4.1 Single-clause induction

In this section we will provide two unprovability results for extensions of a saturation-based system by the single-clause induction rule described in [RV19; Haj+20]. In Section 4.1.1 we introduce the two variants of single-clause induction considered in this section. We will make use of the results from Chapter 3 to obtain a logical upper bound for saturation systems based on single-clause induction. After that, we give two unprovability results for systems based on the single-clause induction rule. In Section 4.1.2 we

give an unprovability result based on the separation of induction over literals and induction over quantifier-free formulas. In Section 4.1.3 provide another independence by making use of an independence result from the literature on mathematical logic. The second unprovability result relies on an independence result for induction over quantifier-free formulas and is therefore more powerful than the first one.

4.1.1 Definition and an upper bound

In [RV19] Reger and Voronkov describe an AITP system that extends a sound saturation-based proof system by various induction rules, among others the single-clause induction rule given below. Here we give a slightly different formulation because it better fits to the more recent formulations of related induction rules.

Definition 4.1.1. *The rigid single-clause induction rule* $\mathrm{RSCIND_S}$ *is given by*

$$\frac{\overline{L}(a) \vee C}{cnf(sk^{\exists}(I_x L(x)))}\ \mathrm{RSCIND_S},$$

where a is a constant symbol, $L(x)$ is a literal free of a*, and $L(a)$ is ground.*

We call the rule rigid because it replaces all the occurrences of a given constant by the induction variable. In this thesis we adopt the designation "single-clause" to indicates that the rule operates on a single clause as premise. This designation is chosen in analogy to the designation of the induction rule rule introduced in [Haj+21b] that we will consider in Section 4.2.

In a practical implementation the rule $\mathrm{RSCIND_S}$ will not apply to every clause of the form $\overline{L}(a) \vee C$ but only when some additional conditions are satisfied (see [RV19, Section 4]). Moreover, for the sake of efficiency the rule is usually implemented in such a way that it immediately resolves the conclusion of the induction axiom $I_x L(x)$ against the clause $\overline{L}(a) \vee C$ (see [RV19, Section 3.2]), thus, effectively deriving the clauses

$$\overline{L}(0) \vee L(c) \vee C,$$
$$\overline{L}(0) \vee \overline{L}(s(c)) \vee C.$$

Our method for providing unprovability results is too coarse to distinguish between these variants, hence we work with the more straightforward to state variant given by Definition 4.1.1.

Remark 4.1.2. *Empirical evidence suggests that generating only the clauses obtained by resolving the induction axiom $I_x L$ with the clause $\overline{L}(a) \vee C$ results in a weaker system. Investigating such variants of the rule* $\mathrm{RSCIND_S}$ *is out of the scope of this thesis and is left as future work.*

It is interesting to observe that the rule $\mathrm{RSCIND_S}$ draws the literals for the induction from the set of already derived clauses and thus does not introduce any new syntactic material into the induction formulas. The motivation for choosing the very restricted

induction rule RSCIND_S is to solve problems that require "little" induction reasoning and complex first-order reasoning [RV19]. In particular the induction rule is chosen so as to not generate too many clauses, which otherwise would potentially result in performance issues. Empirical observations [Haj+20], however, suggest that this method is unable to deal even with very simple yet practically relevant problems such as proving $x+(x+x) = (x+x)+x$ from the usual defining equations for addition. In order to relax the overly restricting analyticity inherent to the rule RSCIND_S, [Haj+20] introduces the following induction rule.

Definition 4.1.3. *The single-clause induction rule* SCIND_S *is given by*

$$\frac{\overline{L}(a) \vee C}{cnf\left(sk^{\exists}(I_x L(x))\right)}\ \text{SCIND}_\text{S}$$

where a is a constant symbol, $L(x)$ is a literal, and the literal $L(a)$ is ground.

This rule reduces the degree of analyticity by allowing the constant symbol a to occur in the induction formula, so that induction may be carried out on slight generalizations of the currently derived literals. This results in more possibilities to add induction axioms to the search space and thus makes search more difficult, but the degree of analyticity of the induction is reduced sufficiently to make the method able to prove some challenge formulas such as for example $x + (x + x) = (x + x) + x$ (see [Haj+20] for details).

Example 4.1.4. *Let $\mathcal{S}$ be a refutationally complete saturation system and let $\varphi(x, y) \coloneqq x + (x + y) = (x + x) + y$. Consider the clause set $\mathcal{C}_0 \coloneqq cnf(sk^{\exists}(\mathcal{T})) \cup \{\neg\varphi(c_0, c_0)\}$, where $sk^{\exists}(\neg(\forall x)\varphi(x, x)) = \varphi(c_0, c_0)$. Then by an application of* SCIND_S *we obtain the clause set $\mathcal{C}_1 \coloneqq \mathcal{C}_0 \cup cnf(sk^{\exists}(I_x\varphi(c_0, x)))$. Since $\mathcal{C}_1 \models \neg\varphi(c_0, c_0)$, we have*

$$\mathcal{C}_1 \models \neg\varphi(c_0, 0) \vee (\varphi(c_0, c_1) \wedge \neg\varphi(c_0, s(c_1)))$$

where $sk^{\exists}(I_x\varphi(c_0, x)) = \varphi(c_0, 0) \wedge (\varphi(c_0, c_1) \to \varphi(c_0, s(c_1)) \to (\forall x)\varphi(c_0, x)$. Now observe that $\mathcal{C}_1 \models c_0 + (c_0 + 0) = c_0 + c_0 = (c_0 + c_0) + 0$. Hence,

$$\mathcal{C}_1 \models \varphi(c_0, c_1) \wedge \neg\varphi(c_0, s(c_1)).$$

Therefore, we have $\mathcal{C}_1 \models c_0 + (c_0 + s(c_1)) = s(c_0 + (c_0 + c_1)) = s((c_0 + c_0) + c_1) = (c_0 + c_0) + s(c_1)$. That is $\mathcal{C}_1 \models \bot$. Hence, by the refutational completeness of $\mathcal{S}$ we obtain a refutation of $\mathcal{C}_1$

Since RSCIND_S is simulated by SCIND_S, we will in the following concentrate on the rule SCIND_S. We will now provide an upper bound on the logical strength of a sound saturation system extended by the rule SCIND_S.

Proposition 4.1.5. *Let $\mathcal{S}$ be a sound saturation system and T a theory. If $cnf(sk^{\exists}(T))$ is refuted by $\mathcal{S} + \text{SCIND}_\text{S}$, then sk^{ω} $(\mathcal{L}(T) \cup \mathcal{L}_0)$-SA $+ T +$ Literal$(\mathcal{L}(sk^{\exists}(T)))$-IND is inconsistent.*

Proof. It suffices to observe that every application of the rule SCIND_S can be replaced by an application of $\text{Literal}(\mathcal{L}(C))\text{-GIND}_\text{S}$ and to apply Lemma 3.2.6. □

Corollary 4.1.6. *Let $\mathcal{S}$ be a sound saturation system and T a Skolem-free $\exists_2$ theory. If $cnf(sk^\exists(T))$ is refuted by $\mathcal{S} + \text{SCIND}_\text{S}$, then $T + \text{Literal}(\mathcal{L}(T))\text{-IND}$ is inconsistent.*

Proof. It suffices to observe that $\mathcal{L}(sk^\exists(T)) = \mathcal{L}(T) \cup \Sigma$, where Σ is a set of nullary function symbols. Hence, $\text{Literal}(\mathcal{L}(T))\text{-IND} \equiv \text{Literal}(\mathcal{L}(sk^\exists(T)))\text{-IND}$ and we can apply Lemma 3.2.8. □

4.1.2 Unprovability by literal induction

In this section we will provide a simple unprovability result for the single-clause induction rule given in Definition 4.1.3. Roughly speaking, the result developed in this section shows that a sound (in the sense of Definition 2.4.10) saturation system extended by the induction rule SCIND_S does not prove that every number is either even or odd. We will obtain this result by making use of the upper bound of Corollary 4.1.6 and by providing a suitable independence result. This result will show us that induction for literals is in general weaker than induction for quantifier-free formulas.

We will work over the following background theory.

Definition 4.1.7. *Let $E/1$ and $O/1$ be predicate symbols, then the theory $\mathcal{A}_1$ extends the base theory of the natural numbers $\mathcal{A}_0$ by the following axioms*

$$E(0), \tag{A1.1}$$
$$E(x) \to O(s(x)), \tag{A1.2}$$
$$O(x) \to E(s(x)). \tag{A1.3}$$

The property that is of interest to us is $E(x) \vee O(x)$. We will show that this formula cannot be proved by induction on literals alone, but of course becomes provable when we consider induction formulas that contain disjunction.

Lemma 4.1.8. $\mathcal{A}_1 + I_x(E(x) \vee O(x)) \vdash E(x) \vee O(x)$.

Proof. Straightforward. □

By a simple model-theoretic construction we will show that induction for literals is not enough to prove $E(x) \vee O(x)$.

Definition 4.1.9. *The structure M_1 consists of pairs (b, n) with $b \in \{0, 1\}$ and $n \in \mathbb{Z}$ such that if $b = 0$, then $n \in \mathbb{N}$. Moreover, the structure M_1 interprets the symbols as follows*

$$0^{M_1} = (0, 0),$$
$$s^{M_1}(b, n) = (b, n + 1),$$
$$E^{M_1} = \{(0, n) \mid n \in \mathbb{N}, n \text{ is even}\},$$
$$O^{M_1} = \{(0, n) \mid n \in \mathbb{N}, n \text{ is odd}\}.$$

Lemma 4.1.10. $M_1 \models \mathcal{A}_1 + \text{Pred} + \text{ACY}$.

Proof. Routine. □

Let us make a simple but crucial observation about the structure M_1.

Lemma 4.1.11. *Let $L(x)$ be a non-equational $\mathcal{L}(\mathcal{A}_1)$ literal containing the variable x, then either $M_1 \not\models L(0)$ or $M_1 \not\models L(x) \to L(s(x))$.*

Proof. We proceed indirectly and assume that $M_1 \models L(0)$ and $M_1 \models L(x) \to L(s(x))$. Then we have $M_1 \models L(\overline{m})$, for all $m \in \mathbb{N}$. We start with the case where L is of the form $E(s^k(x))$. We have $M_1 \models E((0,k))$ and $M_1 \models E((0,k+1))$. However, by the construction of M_1, we either have $(0,k) \notin E^{M_1}$ or $(0,k+1) \notin E^{M_1}$. Contradiction!

Now we consider the case where L is of the form $\neg E(s^k(x))$, for some $k \in \mathbb{N}$. Then, we have $(0,n+k) \notin E^{M_1}$, for all $n \in \mathbb{N}$. In particular $(0,k) \notin E^{M_1}$ and $(0,k+1) \notin E^{M_1}$. However, this is impossible by the construction of M_1. The cases where $L(x)$ is of the form $O(s^k(x))$ or $\neg O(s^k(x))$ are analogous. □

Now we can show that the structure M_1 satisfies literal induction.

Proposition 4.1.12. $M_1 \models \text{Literal}(\mathcal{A}_1)\text{-IND}$.

Proof. Let $L(x,\vec{z})$ be a $\mathcal{L}(\mathcal{A}_1)$ literal and let $\vec{d}$ be elements of M_1. Now assume that $M_1 \models L(0,\vec{d})$ and $M_1 \models L(x,\vec{d}) \to L(s(x),\vec{d})$. The case where L does not contain the variable x is trivial. Hence, by Lemma 4.1.11, L is an equational literal. Therefore, we are done by Corollary 2.6.4. □

We now obtain the following independence result.

Proposition 4.1.13. $\mathcal{A}_1 + \text{Literal}(\mathcal{A}_1)\text{-IND} \nvdash E(x) \vee O(x)$

Proof. By Proposition 4.1.12 it suffices to observe that $(1,0) \notin E^{M_1}$ and $(1,0) \notin O^{M_1}$. □

Since the theory $\mathcal{A}_1 + \exists x\,(\neg E(x) \wedge \neg O(x))$ is $\exists_2$, we obtain an unprovability result for sound saturation systems extended by the rule SCIND_S.

Theorem 4.1.14. *Let $\mathcal{S}$ be a sound saturation system, then the system $\mathcal{S} + \text{SCIND}_\text{S}$ does not refute the clause set $cnf(sk^\exists(\mathcal{A}_1 + \exists x\,(\neg E(x) \wedge \neg O(x))))$.*

Proof. Suppose that $\mathcal{S} + \text{SCIND}_\text{S}$ refutes $cnf(sk^\exists(\mathcal{A}_1 + (\exists x)(\neg E(x) \wedge \neg O(x))))$, then by Corollary 4.1.6, $\mathcal{A}_1 + \text{Literal}(\mathcal{L}(\mathcal{A}_1))\text{-IND} \vdash (\forall x)(E(x) \vee O(x))$. This contradicts Proposition 4.1.13. □

This result is very simple but interesting because it tells us that the absence of disjunction from the induction formulas considered by the single-clause induction rule makes the system unable to prove certain elementary properties involving disjunction. One way to address this unprovability result is to extend the induction rule so as to carry out induction on clauses. Such an improvement of the single-clause induction rule is described by

[Haj21; Haj+21b] and will be considered in Section 4.2. However, there are other possible extensions of induction for literals that could be considered. For example, [Haj+21b, Section IV.B] considers an extension that abstracts some occurrences of terms in literals (or clauses) by universal quantifiers. The following lemma shows that our result covers such an improvement of the rule $\mathrm{SCIND}_{\mathrm{S}}$.

Let Φ be a set of formulas and $k \in \mathbb{N}$, then by $\forall_k \Phi$ we denote the set of formulas of the form $(Q_0 y_0)(Q_1 y_1) \dots (Q_{k-1} y_{k-1})\varphi$, where $\varphi \in \Phi$ and for $i = 0, \dots, k-1$, $Q_i = \forall$ if i is even and $Q_i = \exists$ if i is odd.

Lemma 4.1.15. $\mathcal{A}_1 + \mathrm{Literal}(\mathcal{L}(\mathcal{A}_1))\text{-IND} \vdash \bigcup_{k\in\mathbb{N}} \forall_k \mathrm{Literal}(\mathcal{L}(\mathcal{A}_1))\text{-IND}$.

Proof. Consider the induction axiom $I_x\psi$ where $\psi = (Q_0 y_0) \dots (Q_n y_n) L(x, \vec{y}, \vec{z})$, L is a $\mathcal{L}(\mathcal{A}_1)$ literal and $Q_0, \dots, Q_n \in \{\forall, \exists\}$. If L is an equational literal, then the claim follows from Theorem 2.6.3. If L is of the form $E(t)$, $O(t)$, $\neg E(t)$, or $\neg O(t)$, then we consider two cases. If $t = s^k(x)$, then $\vdash \psi \leftrightarrow L$, thus, $\mathrm{Literal}(\mathcal{L}(\mathcal{A}_1)) \vdash I_x\psi$. Otherwise, L does not contain the induction variable x and therefore $\vdash I_x\psi$. □

We could go on and consider many other possible extensions at this point but this would go beyond the scope of the section.

4.1.3 Unprovabililty by open induction

In the previous sections we have introduced two variants of the single-clause induction rule as defined in [RV19] and [Haj+20]. We have derived an upper bound on the strength of a sound saturation system extended by one of these rules. Finally, we have given an unprovability result that exploits the restriction of the induction formulas to literals. This unprovability result could easily be overcome by considering induction for a larger class of quantifier-free formulas. In this section we will make use of results of [Sho58] to provide more unprovability results for the single-clause induction rule. The results given in this section are based on an independence result for induction over quantifier-free formulas. Thus, the unprovability results of this section are more powerful in the sense that they cannot be overcome in the same way as the unprovability result of Theorem 4.1.14.

The base theory of linear arithmetic together with induction for quantifier-free formulas is sufficient to derive the following four simple facts about addition and the predecessor function.

Lemma 4.1.16. *The theory* $B + \mathrm{Open}(\mathcal{L}_{\mathrm{LA}})\text{-IND}$ *proves the following formulas*

$$x = 0 \vee x = s(p(x)), \tag{B1}$$
$$x + y = y + x, \tag{B2}$$
$$x + (y + z) = (x + y) + z, \tag{B3}$$
$$x + y = x + z \to y = z. \tag{B4}$$

Proof. Routine. □

Definition 4.1.17. *The theory* B' *is axiomatized by* (A1)–(A5) *and* (B1)–(B4).

The following result of Shoenfield shows that in the setting of linear arithmetic the above four formulas are so to speak the only consequences of the induction schema for quantifier-free formulas.

Theorem 4.1.18 ([Sho58])**.** $B + \mathrm{Open}(\mathcal{L}_{\mathrm{LA}})\text{-IND} \equiv B'$.

Proof. See the proof of [Sho58, Theorem 1]. □

This result is interesting from a practical perspective because it replaces an infinite set of axioms by just four axioms. Moreover, this replacement of the induction schema by four natural axioms greatly simplifies model constructions.

Definition 4.1.19. *Let $m, n \in \mathbb{N}$ be natural numbers, then we define*

$$m \cdot x = m \cdot y \to x = y \tag{C_m}$$
$$s^n(m \cdot x) \neq m \cdot y. \tag{$D_{m,n}$}$$

The proof of [Sho58, Theorem 2] can be seen to give the following two independence results. For the sake of completeness we provide the model constructions.

Lemma 4.1.20. *Let $n, m \in \mathbb{N}$ with $0 < n < m$, then $B + \mathrm{Open}(\mathcal{L}(B))\text{-IND} \nvdash D_{m,n}$*

Proof. Let the domain of the structure M_m consist of the elements of the form $(i, \frac{k}{m})$, where $i \in \mathbb{N}$ and $k \in \mathbb{Z}$ such that if $i = 0$, then $k \in \mathbb{N}$. Let the structure M_m interpret the non-logical symbols as follows

$$\begin{aligned}
0^{M_m} &= (0,0),\\
s^{M_m}((i,x)) &= (i, x+1),\\
p^{M_m}((0,0)) &= (0,0),\\
p^{M_m}((i,x)) &= (i, x-1), \text{if } (i,x) \neq (0,0),\\
(i_1,x_1) +^{M_m} (i_2,x_2) &= (i_1+i_2, x_1+x_2).
\end{aligned}$$

By Theorem 4.1.18 it is straightforward to check that $M_m \models B + \mathrm{Open}(\mathcal{L}(B))\text{-IND}$. Now observe that $M_m \models s^n(m \cdot (1,0)) = s^n((1, m\frac{0}{m})) = s^n((1,0)) = (1,n)$ and $M_m \models m \cdot (1, \frac{n}{m}) = (1, m\frac{n}{m}) = (1,n)$. Thus, $M_m \not\models s^n(m \cdot x) \neq m \cdot y$. □

Lemma 4.1.21. *Let $m \in \mathbb{N}$ with $1 < m$, then $\mathcal{T} + \mathrm{Open}(\mathcal{T})\text{-IND} \nvdash C_m$.*

Proof. Let x be a real number, then the fractional part of x is given by $\{x\} \coloneqq x - \lfloor x \rfloor$. For real numbers x_1, x_2 we define $x_1 \dot{+} x_2 \coloneqq \lfloor x_1 \rfloor + \lfloor x_2 \rfloor + \{\{x_1\} + \{x_2\}\}$. Let M'_m be the structure with the same domain as the structure M_m constructed in the proof of Lemma 4.1.20, that interprets the symbols 0, s, p as M_m, and interprets $+$ as follows

$$(i_1,x_1) +^{M'_k} (i_2,x_2) = (i_1+i_2, x_1 \dot{+} x_2).$$

By Theorem 4.1.18 it is routine to check that $M'_m \models B + \mathrm{Open}(\mathcal{L}(B))\text{-IND}$. Now observe that $M'_m \models m \cdot (1, \frac{1}{m}) = \left(1, \sum_{i=1}^{m} \lfloor \frac{1}{m} \rfloor + \left\{\sum_{i=1}^{m} \{\frac{1}{m}\}\right\}\right) = (1,0)$ and $M'_m \models m \cdot (1,0) = (1,0)$. Since $(1,0) \neq (1, \frac{1}{m})$, $M'_m \not\models m \cdot x = m \cdot y \to x = y$. □

From these independence results we immediately obtain unprovability results for sound saturation systems extended by the ground induction rule for quantifier-free formulas over the language of the initial clauses.

Definition 4.1.22. *Let $m, n \in \mathbb{N}$, then the clause sets $\mathcal{C}_m$ and $\mathcal{D}_{m,n}$ are given by*

$$cnf(sk^{\exists}(B' + \neg(\forall x)(\forall y)C_m)), \tag{$\mathcal{C}_m$}$$
$$cnf(sk^{\exists}(B' + \neg(\forall x)(\forall y)D_{m,n})). \tag{$\mathcal{D}_{m,n}$}$$

Theorem 4.1.23. *Let $\mathcal{S}$ be a sound saturation system and $\mathcal{C} \in \{\mathcal{C}_m, \mathcal{D}_{m,n} \mid 0 < n < m\}$, then $\mathcal{S} + \mathrm{Open}(\mathcal{L}(\mathcal{C}))\text{-GIND}_\mathrm{S}$ does not refute the clause set $\mathcal{C}$.*

Proof. We consider the case for $\mathcal{C} = \mathcal{C}_m$ with $1 < m$ and assume that $\mathcal{S}+\mathrm{Open}(\mathcal{L}(\mathcal{C}))\text{-GIND}_\mathrm{S}$ refutes $\mathcal{C}$. Observe that since $B' + \neg(\forall x)(\forall y)C_m$ is $\exists_2$, we have $\mathrm{Open}(\mathcal{L}(\mathcal{C}))\text{-IND} \equiv \mathrm{Open}(\mathcal{L}(B))\text{-IND}$. Hence, by Theorem 3.2.13 we obtain $B + \mathrm{Open}(\mathcal{L}(B))\text{-IND} \vdash C_m$. This contradicts Lemma 4.1.21. The other case is treated analogously. □

Corollary 4.1.24. *Let $\mathcal{S}$ be a sound saturation system and $\mathcal{C} \in \{\mathcal{C}_m, \mathcal{D}_{m,n} \mid 0 < n < m\}$, then $\mathcal{S} + \mathrm{SCIND}_\mathrm{S}$ does not refute the clause set $\mathcal{C}$.*

Proof. An immediate consequence of Theorem 4.1.23 and Corollary 4.1.6. □

The unprovability result gives rise to the question which features a system needs in order to prove the sentences C_m and $D_{m,n}$. In the following we mention some extensions of the open induction schema that would allow us to overcome the unprovability results above. The extensions we suggest are theoretical in the sense that we do not take into account whether they can be implemented efficiently in a saturation system.

A possible extension follows from a remark by Shoenfield [Sho58] that C_m and $D_{m,n}$ can be proved with parameterized double induction (also known as simultaneous induction) on quantifier-free formulas.

Definition 4.1.25. *Let $\gamma(x, y, \vec{z})$ be a formula, then the sentence $I_{(x,y)}\gamma$ is given by*

$$((\forall x)\gamma(x, 0, \vec{z}) \wedge (\forall y)\gamma(0, y, \vec{z}) \wedge (\forall x)(\forall y)(\gamma(x, y, \vec{z}) \to \gamma(s(x), s(y), \vec{z}))) \to (\forall x)(\forall y)\gamma(x, y, \vec{z}).$$

Let Φ be a set of formulas, then the double induction schema Φ-DIND for Φ formulas is given by $\Phi\text{-DIND} := \{(\forall \vec{z})I_{(x,y)}\gamma \mid \gamma(x, y, \vec{z}) \in \Phi\}$.

Lemma 4.1.26. *The theory $B + \mathrm{Open}(\mathcal{L}(B))\text{-DIND}$ proves the formulas C_m for $0 < m$ and $D_{m,n}$ for $0 < n < m$.*

Proof. For C_m we proceed by induction on the formula C_m. We consider only one of the base cases, since the other one is symmetric. Assume $m \cdot x = m \cdot 0$. We have to show $x = 0$. Hence, $m \cdot x = 0$. Now suppose that $x \neq 0$, then $x = s(p(x))$ and we have $0 = m \cdot x = s^m(m \cdot p(x))$. This is a contradiction! For the induction step assume $m \cdot x = m \cdot y \to x = y$ and $m \cdot s(x) = m \cdot s(y)$. We have $m \cdot x + \overline{m} = m \cdot s(x) = m \cdot s(y) =$

$m \cdot y + \overline{m}$. Hence $m \cdot x = m \cdot y$, thus $x = y$. Therefore $s(x) = s(y)$. For $D_{m,n}$ we proceed analogously. □

We shall later encounter the extension to simultaneous induction in Section 4.2 where we will consider the multi-clause induction rule.

Another possible extension to overcome the independence of Lemmas 4.1.20 and 4.1.21 is to extend the induction rule used by the system at least to parameter-free $\forall_1$ induction.

Lemma 4.1.27. $B + \forall_1(\mathcal{L}(B))\text{-IND}^- \vdash \text{Open}(\mathcal{L}(B))\text{-IND}$.

Proof. Straightforward via Theorem 4.1.18. □

Lemma 4.1.28. *The theory* $B + \forall_1(\mathcal{L}(B))\text{-IND}^-$ *proves the formulas* C_m *for* $0 < m$ *and* $D_{m,n}$ *for* $0 < n < m$.

Proof. For C_m with $0 < m$ we work in $B + \forall_1(\mathcal{L}(B))\text{-IND}^-$ and proceed by induction on the formula $(\forall y)(n \cdot x = n \cdot y \to x = y)$. By Lemma 4.1.27 we have (B1)–(B4). For the base case we have to show that $n \cdot 0 = n \cdot y \to 0 = y$. By Lemma 2.6.11 we have $n \cdot 0 = 0$. By (B1) we need to distinguish two cases. If $y = 0$, then we are done, otherwise we obtain a contradiction by Lemma 2.6.9.(i). For the induction step we assume $(\forall y)(n \cdot x = n \cdot y \to x = y)$ and $n \cdot s(x) = n \cdot y$. We want to obtain $s(x) = y$. By (A5) and (B2) we obtain $s^n(n \cdot x) = n \cdot s(x) = n \cdot y$. By (B1) we can distinguish two cases. If $y = 0$, then $s^n(n \cdot x) = 0$, which contradicts Lemma 2.6.9.(i). Otherwise $y = s(p(y))$, hence $s^n(n \cdot x) = n \cdot s(x) = n \cdot s(p(y)) = s^n(n \cdot y)$. Therefore by injectivity of s we obtain $n \cdot x = n \cdot p(y)$. Hence, $x = p(y)$ by the induction hypothesis. Thus $s(x) = s(p(y)) = y$. For $D_{m,n}$ with $0 < n < m$ we proceed analogously. □

These observations give rise to the question whether both extensions are equally powerful.

Question 4.1.29. *Let* T *be a theory. Do the theories* $T + \forall_1(\mathcal{L}(T))\text{-IND}^-$ *and* $T + \text{Open}(\mathcal{L}(T))\text{-DIND}$ *have the same* $\forall_1(\mathcal{L}(T))$ *consequences?*

Answering this question (negatively) is probably not very difficult but we leave the question open since this would go beyond the purpose of the section. For the setting of linear arithmetic the following theorem of Shoenfield and Lemmas 4.1.26 and 4.1.28 provide an answer.

Theorem 4.1.30 ([Sho58, Theorem 3])**.** $B' + \{C_m \mid 0 < m\} + \{D_{m,n} \mid 0 < n < m\}$ *is complete for* $\forall_1(\mathcal{L}(B))$ *sentences.*

Hence, it follows that at least in the setting of linear arithmetic double induction over quantifier-free formulas and parameter-free $\forall_1$ induction are sufficient to prove all true quantifier-free formulas. Let us mention again that the extensions of induction mechanisms that introduce universal quantification into induction formulas are discussed in [Haj+21b].

Let us now briefly discuss how relevant the unprovability result of Corollary 4.1.24 is for induction on literals. The underlying independence results rely on an independence

from induction for quantifier-free induction rather than on the much weaker induction for literals. Hence, the result may seem quite unrelated to literal induction. However, the following shows this result can be reformulated as a result about literal induction.

Lemma 4.1.31. *The theory* $B + \mathrm{Literal}(\mathcal{L}(B))$-IND *proves* B1–B4.

Proof. Proving B2 and B3 is straightforward. For B4 we show the contrapositive $y \neq z \to x + y \neq x + z$. We assume $y \neq z$ and proceed by induction on x in the formula $x + y \neq x + z$. For the base case we have to show $0 + y \neq 0 + z$. By B2 and the definition of $+$ the formula $0 + y \neq 0 + z$ is equivalent to $y \neq z$ which we have assumed. For the induction step we assume $s(x)+y \neq s(x)+z$. By B2 and A5 we obtain $s(x+y) \neq s(x+z)$, hence $x + y \neq x + z$ and we are done.

Proving B1 is slightly more complicated because the induction interacts even more with the context. We assume $x \neq 0$ and we have to show $x = s(p(x))$. We proceed by induction on y in the formula $x \neq y$. The induction base is trivial since we have assumed $x \neq 0$. For the induction step we assume $x \neq y_0$ and we have to show $x \neq s(y_0)$. Hence we assume $x = s(y_0)$. Now we have $s(p(x)) = s(p(s(y_0))) = s(y_0) = x$ and we are done. Therefore we obtain the formula $(\forall y)x \neq y$ and in particular $x \neq x$, which is a contradiction. Hence we obtain $x = s(p(x))$. □

In the light of Shoenfield's theorem it is now clear that induction for literals is as powerful as quantifier-free induction.

Proposition 4.1.32. $B + \mathrm{Literal}(B)$-IND $\equiv B + \mathrm{Open}(B)$-IND.

Proof. The direction from right to left is obvious. For the direction from left to right follows from Lemma 4.1.31 and Theorem 4.1.18. □

Therefore, by the above, it is, in the setting of linear arithmetic, not possible to improve the unprovability results of this section by taking into account the restriction of the induction to literals. In other words, we cannot find an unprovability result that relies on a separation of $B + \mathrm{Open}(\mathcal{L}(B))$-IND and $B + \mathrm{Literal}(\mathcal{L}(B))$-IND.

In a similar way to what we did in this section we obtain many more unprovability results by using independence results of Shepherdson [She64] and Schmerl [Sch88]. However, these results are less elementary in two ways. Firstly, they take place in a language that besides the symbols of linear arithmetic contains the symbol $\dot{-}/2$ for truncated subtraction and $\cdot/2$ for multiplication. Secondly, the properties that are shown independent of the base theory with open induction express more complicated properties such as the irrationality of the square root of two, Fermat's last theorem for $n = 3$, and other Diophantine equations. Hence, these independence results are currently not practically as relevant as the independence results stated above.

4.2 Multi-clause induction

Recently a more powerful induction rule was introduced in [Haj21; Haj+21b] that operates on multiple clauses. In this section we will consider some practically realistic

instances of this rule. Section 4.2.1 defines a variant of the multi-clause induction rule that includes some of the extensions suggested by the unprovability results of Section 4.1 and provides an upper bound. In Section 4.2.2 we make use of the upper bound to give a simple unprovability result. Finally, we consider in Section 4.2.3 a slightly stronger variant of the multi-clause induction which overcomes the unprovability result given in Section 4.2.2 and we provide another unprovability result.

4.2.1 Definition and upper bound

The multi-clause induction rule introduced in [Haj21; Haj+21b] improves upon the single-clause induction rule in that the rule performs induction on clauses by selecting literals from multiple clauses. This improvement of the single-clause induction rule is based on empirical observations, see [Haj+21b, Section IV]. Our results of the previous section complement such observations and highlight the necessity to consider stronger rules.

In this section we consider an instance of the multi-clause induction rule given in [Haj+21b] that provides induction on several variables simultaneously. This is according to the improvement suggested by Lemma 4.1.26. This form of induction has some of the strength of induction for formulas with universal quantification. There are several ways to define multivariate induction. Here we consider a simple variant that corresponds to a straightforward generalization of the induction schema of Definition 4.1.25. This variant of simultaneous induction also appears in Cruanes' system [Cru17], which we will consider in Chapter 5. Hence this method seems to be of practical interest. Let us note that [Haj+21b] considers an even stronger variant that allows for universal quantification over clauses and Lemma 4.1.28 suggests a similar improvement. In this thesis we will not consider these stronger variants and leave their investigation for future work.

Let us start by introducing some notation for dealing with finite sequences of terms. Let $\vec{t} = (t_1, \ldots, t_n)$ be a finite sequence of terms, then we write $s(\vec{t})$ for the vector of terms $(s(t_1), \ldots, s(t_n))$. Let $\vec{x} = (x_1, \ldots, x_n)$ be a finite sequence and $i \in \mathbb{N}$ with $1 \leq i \leq n$, then we define $\vec{x}_{<i} = (x_1, \ldots, x_{i-1})$ and $\vec{x}_{>i} = (x_{i+1}, \ldots, x_n)$.

Definition 4.2.1 (Diagonal Induction)**.** *Let $\varphi(\vec{x}, \vec{z})$ be a formula, then the formula $I^{\mathrm{Diag}}_{\vec{x}}\varphi$ is given by*

$$\left(\bigwedge_{i=1}^{|\vec{x}|}(\forall\vec{x}_{<i})(\forall\vec{x}_{>i})\varphi(\vec{x}_{<i}, 0, \vec{x}_{>i}, \vec{z}) \wedge (\forall\vec{x})(\varphi(\vec{x}, \vec{z}) \to \varphi(s(\vec{x}), \vec{z}))\right) \to (\forall\vec{x})\varphi.$$

Let $m \in \mathbb{N}$ and Φ a set of formulas, then the m-variate diagonal induction schema Φ-Diag$_m$IND is axiomatized by the universal closure of the formulas $I^{\mathrm{Diag}}_{\vec{x}}\varphi$ where $\vec{x}$ is an a sequence of m variables and $\varphi \in \Phi$. Moreover, we define Φ-Diag$_\omega$IND $:=$ $\bigcup_{m<\omega} \Phi$-Diag$_m$IND.

Remark 4.2.2. *A stronger form of simultaneous induction can be considered by allowing for induction hypotheses of the form $\varphi(\vec{t}, \vec{z})$, where $\vec{t} = (t_1, \ldots, t_m)$ and $t_i = x_i$ or $t_i = s(x_i)$ and $\vec{t} \neq s(\vec{x})$. This stronger form of simultaneous induction is used in the HIP*

induction prover, see [Ros12]. However, we will not deal with this induction principle in this thesis.

Whenever considering a variant of an induction schema it is useful to justify the schema in terms of the structural induction schema, because this provides us with inductive soundness and an idea of the possible strength of the schema at hand.

Lemma 4.2.3. *Let $\varphi(\vec{x}, \vec{z})$ be a formula, then*

$$\mathrm{Pred} \vdash I_{x_1}(\forall \vec{x}_{>1})\varphi \to I^{\mathrm{Diag}}_{\vec{x}}\varphi.$$

Proof. Let $|\vec{x}| = n$ and assume

$$\bigwedge_{i=1}^{n} (\forall \vec{x}_{<i})(\forall \vec{x}_{>i})\varphi(\vec{x}_{<i}, 0, \vec{x}_{>i}, \vec{z}), \tag{$*$}$$

$$(\forall \vec{x})(\varphi(\vec{x}, \vec{z}) \to \varphi(s(\vec{x}), \vec{z})). \tag{$\star$}$$

For the induction base we have $(\forall \vec{x}_{>1})\varphi(0, \vec{x}_{>1}, \vec{z})$ by $(*)$. For the induction step assume $(\forall \vec{x}_{>1})\varphi(x_1, \vec{x}_{>1}, \vec{z})$ and let $\vec{x}_{>1}$ be arbitrary but fixed. If x_i for some $i \in \{2, \ldots, n\}$, then we are done by $(*)$. Otherwise, for $i = 2, \ldots, n$ there exists x'_i such that $x_i = s(x'_i)$. By the induction hypothesis we have $\varphi(x_1, x'_2, \ldots, x'_n, \vec{z})$, hence by $(\star)$ we obtain $\varphi(s(x_1), s(x'_2), \ldots, s(x'_n), \vec{z})$, that is, $\varphi(s(x_1), \vec{x}_{>1}, \vec{z})$. □

Thus, diagonal induction is just a restricted form of universal quantification over the induction formulas. We are now ready to give the definition of the multi-clause induction rule that we will consider in the following.

Definition 4.2.4. *The multi-clause induction rule* $\mathrm{MCIND}_{\mathrm{S}}$ *is given by*

$$\frac{\overline{L_1}(\vec{t}) \vee C_1 \quad \ldots \quad \overline{L_n}(\vec{t}) \vee C_n}{cnf\left(sk^{\exists}\left(I^{\mathrm{Diag}}_{\vec{x}} \bigvee_{i=1}^{n} L_i(\vec{x})\right)\right)}\ \mathrm{MCIND}_{\mathrm{S}},$$

where $L_1(\vec{x})$, …, $L_n(\vec{x})$ are literals, $\vec{t}$ is a finite sequence of ground terms, and C_1, …, C_n are clauses.

We observe that this rule is powerful enough to overcome the unprovability results of Theorem 4.1.14 and Corollary 4.1.24.

Example 4.2.5. *Consider the clause set $\mathcal{C} \coloneqq cnf(\mathcal{A}_1) \cup \{\neg E(c), \neg O(c)\}$, where*

$$sk^{\exists}((\exists x)(\neg E(x) \wedge \neg O(x))) = \neg E(c) \wedge \neg O(c).$$

Since $\mathcal{C}$ contains the clauses $\neg E(c)$ and $\neg O(c)$, the rule $\mathrm{MCIND}_{\mathrm{S}}$ *may be applied to derive the clauses*

$$cnf(sk^{\exists}(I_x(E(x) \vee O(x)))).$$

By Lemma 4.1.8 the resulting clause set is unsatisfiable.

Example 4.2.6. *Let $m \in \mathbb{N}$ with $m > 1$ and $\mathcal{C}_0 \coloneqq cnf(B') \cup \{m \cdot c_0 = m \cdot c_1, c_0 \neq c_1\}$, where $sk^{\forall}((\forall x)(\forall y)C_m) = C_m(c_0, c_1)$. Then an application of the rule* MCIND_S *gives us the clause set $\mathcal{C}_1 = \mathcal{C}_0 \cup cnf(sk^{\exists}(I^{\text{Diag}}_{(x_1,x_2)}(m \cdot x_1 \neq m \cdot x_2 \vee x_1 = x_2)))$. It is straightforward to see that this clause set is unsatisfiable. Hence a refutationally complete saturation system refutes $\mathcal{C}_1$.*

We will now derive an upper bound on the strength for the multi-clause induction rule given above. Since this rule uses induction axioms that are not of the form of the induction axioms given in Definition 2.5.9, we cannot directly use the results obtained so far in order to obtain an upper bound. However, given the definition of the rule MCIND_S and the analogous situations considered so far (see Theorem 3.2.13 and Proposition 4.1.5) it is straightforward to obtain an upper bound.

Proposition 4.2.7. *Let $\mathcal{S}$ be a sound saturation system and T a theory. If $\mathcal{S} + \text{MCIND}_\text{S}$ refutes the clause set $cnf(sk^{\exists}(T))$, then the theory*

$$sk^{\omega}\,(\mathcal{L}(T) \cup \mathcal{L}_0)\text{-SA} + T + \text{Clause}(\mathcal{L}(sk^{\exists}(T)))\text{-Diag}_{\omega}\text{IND}$$

is inconsistent.

Proof. Let $\mathcal{C}_0, \ldots, \mathcal{C}_n$ a $\mathcal{S} + \text{MCIND}_\text{S}$ deduction, then we proceed by induction on n and show the slightly stronger claim that there exists a set of nullary function symbols Σ_n such that $\mathcal{L}(\mathcal{C}_n) \subseteq \mathcal{L}(\mathcal{C}_0) \cup \Sigma_n \subseteq \mathcal{L}'$ and

$$\mathcal{L}'\text{-SA} + T + \text{Clause}(\mathcal{L}(\mathcal{C}_0))\text{-Diag}_{\omega}\text{IND} \models \mathcal{C}_n,$$

where $\mathcal{L}' = sk^{\omega}\,(\mathcal{L}(\mathcal{C}_0) \cup \mathcal{L}_0)$. The base case $n = 0$ is trivial. For the induction step assume that $\mathcal{L}'\text{-SA} + \mathcal{C}_0 + \text{Clause}(\mathcal{L}(\mathcal{C}_0))\text{-Diag}_{\omega}\text{IND} \models \mathcal{C}_n$. If $\mathcal{C}_{n+1}$ is obtained from $\mathcal{C}_n$ by an inference from $\mathcal{S}$, then by the soundness of $\mathcal{S}$ we are done. If $\mathcal{C}_{n+1}$ is obtained from $\mathcal{C}_n$ by an application of the MCIND_S, then

$$\mathcal{C}_{n+1} = \mathcal{C}_n \cup cnf(sk^{\exists}(I_x \bigvee_{i=1}^{n} L_i(\vec{x}))).$$

By the induction hypothesis there exist $\mathcal{L}(\mathcal{C}_0)$ literals $L'_1(\vec{x}, \vec{y}), \ldots, L'_n(\vec{x}, \vec{y})$ and a finite sequence of nullary function symbols $\vec{c} = (c_1, \ldots, c_{|\vec{y}|})$ with $c_i \in \mathcal{L}'$ for $i = 1, \ldots, |\vec{y}|$ such that $L_i = L'_i(\vec{x}, \vec{c})$. Hence, we have

$$\text{Clause}(\mathcal{L}(\mathcal{C}_0))\text{-Diag}_{\omega}\text{IND} \vdash I^{\text{Diag}}_{\vec{x}} L(\vec{x}, \vec{c}),$$

and furthermore, $\mathcal{L}'\text{-SA}+\text{Clause}(\mathcal{L}(\mathcal{C}_0))\text{-Diag}_{\omega}\text{IND} \vdash sk^{\exists}(I^{\text{Diag}}_{\vec{x}} L(\vec{x}, \vec{c}))$. Since, $I^{\text{Diag}}_{\vec{x}} \bigvee_{i=1}^{n} L_i$ is parameter-free we have $\mathcal{L}(\mathcal{C}_{n+1}) = \mathcal{L}(\mathcal{C}_n) \cup \{d_1, \ldots, d_{|\vec{x}|}\}$, where

$$sk^{\forall}\left((\forall \vec{x})\left(\bigvee_{i=1}^{n} L(\vec{x}) \to \bigvee_{i=1}^{n} L(s(\vec{x}))\right)\right) = \bigvee_{i=1}^{n} L(\vec{d}) \to \bigvee_{i=1}^{n} L(s(\vec{d}))$$

Hence, $\mathcal{L}(\mathcal{C}_{n+1}) \subseteq \mathcal{L}(C_0) \cup \Sigma_n \cup \{d_1, \ldots, d_{|\vec{d}|}\} \subseteq \mathcal{L}'$ and moreover

$$\mathcal{L}'\text{-SA} + \mathcal{C}_0 + \text{Clause}(\mathcal{L}(C_0))\text{-Diag}_\omega\text{IND} \models \mathcal{C}_{n+1}. \qquad \square$$

In other words the multi-clause induction rule given in Definition 4.2.4 corresponds to parameterized induction over clauses. In the $\exists_2$ setting we even get a Skolem-free upper bound.

Corollary 4.2.8. *Let $\mathcal{S}$ be a sound saturation system and T a Skolem-free $\exists_2$ theory. If $\mathcal{S} + \text{MCIND}_\text{S}$ refutes $cnf(sk^\exists(T))$, then the theory $T + \text{Clause}(\mathcal{L}(T))\text{-Diag}_\omega\text{IND}$ is inconsistent.*

Proof. Assume that $\mathcal{S} + \text{MCIND}_\text{S}$ refutes $cnf(sk^\exists(T))$, then by Proposition 4.2.7, the theory $sk^\omega\ (\mathcal{L}(T) \cup \mathcal{L}_0)\text{-SA} + T + \text{Clause}(\mathcal{L}(sk^\exists(T)))\text{-Diag}_\omega\text{IND}$ is inconsistent. Observe that for $\exists_2$ theory we have

$$\text{Clause}(\mathcal{L}(sk^\exists(T)))\text{-Diag}_\omega\text{IND} \equiv \text{Clause}(\mathcal{L}(T))\text{-Diag}_\omega\text{IND}.$$

Hence, $sk^\omega\ (\mathcal{L}(T) \cup \mathcal{L}_0)\text{-SA} + T + \text{Clause}(\mathcal{L}(T))\text{-Diag}_\omega\text{IND}$ is inconsistent. Since T is Skolem-free, the theory $T + \mathcal{L}(T)\text{-Diag}_\omega\text{IND}$ is Skolem-free. Hence, by Proposition 2.2.11 the theory $T + \mathcal{L}(T)\text{-Diag}_\omega\text{IND}$ is inconsistent. $\square$

4.2.2 Unprovability by induction on clauses

In this section we will provide a simple independence result for simultaneous induction over clauses. This independence result together with the upper bound of Corollary 4.2.8 then gives us an unprovability result for the variant of multi-clause induction given in Definition 4.2.4.

We will show that induction for clauses is not powerful enough to prove some conjunctive properties.

Definition 4.2.9. *Let $A/1$ and $B/1$ be predicate symbols, then the theory $\mathcal{A}_2$ extends the theory base theory of the natural numbers $\mathcal{A}_0$ by the universal closure of the following formulas*

$$A(0), \tag{A2.1}$$
$$B(0), \tag{A2.2}$$
$$A(x) \to B(s(x)), \tag{A2.3}$$
$$B(x) \to A(s(x)). \tag{A2.4}$$

The formula $A(x) \wedge B(x)$ is inductive in $\mathcal{A}_2$ and is therefore easily proved.

Lemma 4.2.10. $\mathcal{A}_2 + I_x(A(x) \wedge B(x)) \vdash A(x) \wedge B(x)$.

Proof. Straightforward. $\square$

Remark 4.2.11. *In the light of Theorem 2.6.3 it is easy to see that the theory* $\mathcal{A}_0 + \mathrm{Pred} + \mathrm{ACY} + (\forall x)A(x) + (\forall x)B(x)$ *is complete.*

In the following we will show that induction for clauses does not allow us to prove $A(x) \wedge B(x)$. We will as usual construct a suitable model of $\mathcal{A}_2 + \mathrm{Clause}(\mathcal{L}(\mathcal{A}_2))$-IND.

Definition 4.2.12. *Let* M_2 *be the* $\mathcal{L}(\mathcal{A}_2)$ *structure consisting of pairs of the form* (i, n) *where* $i \in \{1, 0\}$ *and* $n \in \mathbb{Z}$ *such that* $i = 0$ *implies* $n \in \mathbb{N}$. *Furthermore let* M_2 *interpret the symbols as follows*

$$\begin{aligned} 0^{m} &= (0, 0), \\ s^{M_2}((i, n)) &= (i, n + 1), \\ A^{M_2} &= \{(0, n) \mid n \in \mathbb{N}\} \cup \{(1, n) \mid n \in \mathbb{Z}, n \text{ is even}\}, \\ B^{M_2} &= \{(0, n) \mid n \in \mathbb{N}\} \cup \{(1, n) \mid n \in \mathbb{Z}, n \text{ is odd}\} \end{aligned}$$

Lemma 4.2.13. $M_2 \models \mathcal{A}_2 + \mathrm{Pred} + \mathrm{ACY}$.

Proof. Straightforward. □

We will now show that M_2 is also a model of diagonal induction over $\mathcal{L}(\mathcal{A}_2)$ clauses.

Proposition 4.2.14. $M_2 \models \mathrm{Clause}(\mathcal{A}_2)\text{-Diag}_{\omega}\mathrm{IND}$.

Proof. Let $C(\vec{x}, \vec{z})$ be an $\mathcal{L}(\mathcal{A}_2)$ clause and let $\vec{c}$ be a finite sequence of elements of M_2 such that $|\vec{c}| = |\vec{z}|$. Now we assume that

$$M_2 \models \bigwedge_{i=1}^{|\vec{x}|} (\forall \vec{x}_{<i})(\forall \vec{x}_{>i}) C(\vec{x}_{<i}, 0, \vec{x}_{>i}, \vec{c}), \tag{$*$}$$

$$M_2 \models C(\vec{x}, \vec{c}) \to C(s(\vec{x}), \vec{c}). \tag{$\star$}$$

By Lemma 4.2.13 and Corollary 2.6.4 it suffices to consider the case where C contains a non-equational literal containing an occurrence of an induction variable. Let $\vec{d} \in \mathcal{D}(M_2)^{|\vec{x}|}$ with $d_i = (b_i, m_i)$. We will show that $M_2 \models C(\vec{d}, \vec{c})$. Let $M = \{m_i \in \mathbb{N} \mid b_i = 0, i = 1, \ldots, |\vec{x}|\}$. Now we have to consider two cases.

If $M \neq \varnothing$, then let m be the least element of M, then there exists $i_0 \in \{1, \ldots, |\vec{x}|\}$ such that $d_{i_0} = (0, m)$. Let $d'_j := (b_j, m_j - m)$ for $j = 1, \ldots, |\vec{x}|$. Observe that by the choice of m we have $d_j \in \mathcal{D}(M_2)$. By $(*)$ we have $M_2 \models C(\vec{d'}, \vec{c})$. By a straightforward induction and applying $(\star)$ we now obtain $M_2 \models C(\vec{d}, \vec{c})$.

Now let us consider the case where $M = \varnothing$, that is, each element of $\vec{d}$ is a non-standard element. Assume that C contains a literal of the form $A(s^k(x_i))$, where $k \in \mathbb{N}$. If $m_i + k$ is even, then we have $(1, m_i + k) \in A^{M_2}$, hence $M_2 \models A(s^k(d_i))$ and therefore $M_2 \models C(\vec{d}, \vec{c})$. Otherwise, if $m_i + k$ is odd, then $m_i - 1 + k$ is even, $(1, m_i - 1) \in M_2$, and we have $(1, m_i - 1 + k) \in A^{M_2}$. Hence, $M_2 \models C((1, m_1 - 1), \ldots, (1, m_{|\vec{x}|} - 1), \vec{c})$ and therefore by applying the assumption corresponding to the induction step once, we obtain $M_2 \models C(\vec{d}, \vec{c})$.

Assume that C contains a literal of the form $\neg A(s^k(x_i))$, where $k \in \mathbb{N}$. If $m_i + k$ is odd, then we have $(1, m_i + k) \notin A^{M_2}$ and therefore $M_2 \models \neg A(s^k(d_i))$. Hence, $M_2 \models C(\vec{d}, \vec{c})$. If $m_i + k$ is even, then $m_i - 1 + k$ is odd and we have $(1, m_i - 1 + k) \notin A^{M_2}$. Hence, we have $M_2 \models \neg A(s^k((1, m_i - 1)))$ and therefore $M_2 \models C((1, m_1 - 1), \ldots, (1, m_{|\vec{x}|} - 1), \vec{c})$. By applying the assumption $(\star)$ once, we obtain $M_2 \models C(\vec{d}, \vec{c})$.

We proceed analogously if C contains a literal $B(s^k(x))$ or $\neg B(s^k(x))$. □

As an immediate consequence of the proposition above we obtain the following independence result.

Proposition 4.2.15. $\mathcal{A}_2 + \text{Clause}(\mathcal{A}_2)\text{-Diag}_\omega\text{IND} \nvdash A(x)$.

Proof. It suffices to observe that $M_2 \not\models A((1, 1))$. □

This readily gives us an unprovability result for sound saturation systems extended by the multi-clause induction rule of Definition 4.2.4.

Theorem 4.2.16. *Let $\mathcal{S}$ be a sound saturation system, then $\mathcal{S}+\text{MCIND}_\text{S}$ does not refute the clause set $cnf(sk^\exists(\mathcal{A}_2 + (\exists x)\neg A(x)))$.*

Proof. Proceed indirectly by assuming that $\mathcal{S}+\text{MCIND}_\text{S}$ refutes $cnf(sk^\exists(\mathcal{A}_2+(\exists x)\neg A(x)))$. Since $\mathcal{A}_2 + (\exists x)\neg A(x)$ is Skolem-free and $\exists_2$ we can apply Corollary 4.2.8 to obtain the inconsistency of $\mathcal{A}_2 + (\exists x)\neg A(x)$. This contradicts Proposition 4.2.15. □

Let us now discuss this unprovability result. For the unprovability result above we could also have used the formula $A(x) \wedge B(x)$. The reason we have used $A(x)$ is that proving this formula from $\mathcal{A}_2$ by induction is more difficult in the sense that we cannot proceed by induction on the formula $A(x)$ itself. In particular, we have the following interesting result.

Definition 4.2.17. *The sequent calculus induction rule* Ind *is given by*

$$\frac{\Gamma \Longrightarrow \Delta, F(0) \qquad \Gamma, F(\alpha) \Longrightarrow \Delta, F(s(\alpha))}{\Gamma \Longrightarrow \Delta, F(t),} \text{ Ind}$$

where $F(x)$ is a formula, t a term, and the variable α does not occur freely in $F(0), \Gamma$, or Δ (but α may appear in t). By $\mathbf{G} + \text{Ind}$ we denote the proof system obtained by adding the induction rule Ind *to* $\mathbf{G}$.

Proposition 4.2.18. *The sequent $\mathcal{A}_2 \Rightarrow A(x)$ does not have $\mathbf{G} + \text{Ind}$ proof all of whose cuts are atomic.*

Proof. We proceed by induction on the structure of a $\mathbf{G} + \text{Ind}$ proof in ACNF of the sequent $\mathcal{A}_2 \Longrightarrow A(x)$ and show that all the formulas appearing in the succedent are clauses. Hence, proceeding indirectly and assuming there is such a proof, we can extract the induction formulas and obtain $\mathcal{A}_2 + \text{Clause}(\mathcal{L}(\mathcal{A}_2))\text{-IND} \vdash A(x)$. This contradicts Proposition 4.2.15. □

Proposition 4.2.18, thus, provides a very simple non-analyticity result. In other words it does not suffice to reuse the formulas appearing during proof search for induction without at least transforming them. Even though many AITP systems have different analyticity properties and moreover include heuristics, we expect that proving $A(x)$ from $\mathcal{A}_2$ may be challenging for analytic AITP systems.

On the other hand, the unprovability of Theorem 4.2.16 can be overcome very easily by considering induction with larger steps.

Definition 4.2.19. *Let $\varphi(x, \vec{z})$ be a formula and m a natural number, then the weak big-step induction axiom $I^{\mathrm{w}}_{x,m}\varphi$ is given by*

$$\bigwedge_{k=0}^{m} \varphi(\overline{k}, \vec{z}) \wedge \forall x \left(\varphi(x, \vec{z}) \to \varphi(s^{m+1}(x), \vec{z})\right) \to \forall x \varphi(x, \vec{z}).$$

Let Φ be a set of formulas, then Φ-wBSIND is axiomatized by the universal closure of the formulas $I^{\mathrm{w}}_{x,m}\varphi$, where $\varphi \in \Phi$ and m is a natural number.

Remark 4.2.20. *Let $\varphi(x, \vec{z})$ be a formula and $m \in \mathbb{N}$, then the formula $(\forall \vec{z}) I^{\mathrm{w}}_{x,m}\varphi$ is called $\mathrm{I}^{(m+1)}_{x}\varphi$ in [HW18].*

The justification of big-step induction as defined above is later carried out as part of the justification of a slightly more complicated induction schema (see Lemma 4.2.27).

Lemma 4.2.21. $\mathcal{A}_2 + I^{\mathrm{w}}_{x,1}A(x) + I^{\mathrm{w}}_{x,1}B(x) \vdash A(x) \wedge B(x)$.

Proof. By (A2.2) and (A2.4) we obtain $A(\overline{1})$. Moreover, by (A2.3) and (A2.4) we obtain $A(x) \to A(s^2(x))$. Hence, we by $I^{\mathrm{w}}_{x,1}A(x)$ obtain $A(x)$. Analogously, we obtain $B(x)$. □

The above result shows that big-step induction has some of the power of induction over formulas containing conjunction. However, in the following section we will provide an unprovability result that shows that extending the multi-clause induction rule by big-step induction still is not as powerful as induction for formulas with conjunction.

In the light of Lemma 4.2.3 it is natural to ask whether the independence of Proposition 4.2.15 can be overcome by using induction over quantified clauses instead of diagonal induction for clauses.

Question 4.2.22. *Is there a $k \in \mathbb{N}$ such that $\mathcal{A}_2 + \forall_k(\mathrm{Clause}(\mathcal{L}(\mathcal{A}_2)))$-IND $\vdash A(x) \wedge B(x)$?*

Providing a negative answer to the question would also yield an unprovability result for the extension discussed in [Haj+21b, Section IV.B].

4.2.3 Unprovability by big-step induction over clauses

In this section we consider a straightforward extension of the multi-clause induction rule by big-step induction and we provide a simple unprovability result that exploits the absence of conjunction from the induction formulas. The usage of big-step induction

is in particular considered in [Haj+21b] and also appears in the methods considered in Chapters 5 and 6.

Let us start by introducing the big-step induction schema that we will consider in this section. Let $\vec{m} = (m_1, \ldots, m_n)$ be a sequence of natural numbers and let $\vec{t} = (t_1, \ldots, t_n)$ be a sequence of terms, then $s^{\vec{m}}(\vec{t})$ denotes the sequence of terms $(s^{m_1}(t_1), \ldots, s^{m_n}(t_n))$.

Definition 4.2.23. *Let $\varphi(\vec{x}, \vec{z})$ be a formula and $\vec{m}$ a vector of natural numbers with $|\vec{m}| = |\vec{x}|$, then the formula $I^{\mathrm{Diag}}_{\vec{x},\vec{m}}\varphi$ is given by*

$$\left(\bigwedge_{i=1}^{|x|}\bigwedge_{j=0}^{m_j}(\forall\vec{x}_{<i})(\forall\vec{x}_{>i})\varphi(\vec{x}_{<i}, \overline{j}, \vec{x}_{>i}, \vec{z}) \wedge (\forall\vec{x})\Big(\varphi(\vec{x}, \vec{z}) \to \varphi(s(s^{\vec{m}}(\vec{x})), \vec{z})\Big)\right) \to (\forall\vec{x})\varphi.$$

Let Φ be a set of formulas, then the theory Φ-$\mathrm{BS}_{\mathrm{het}}\mathrm{Diag}_\omega\mathrm{IND}$ is axiomatized by the universal closure of the formulas $I^{\mathrm{Diag}}_{\vec{x},\vec{m}}\varphi$ where $\vec{m}$ is a vector of natural numbers $|\vec{m}| = |\vec{x}|$ and $\varphi(\vec{x}, \vec{z}) \in \Phi$.

The induction schema introduced above includes the weak big-step induction schema and the diagonal induction schema considered in the previous sections. Let, for example, $\varphi(x, \vec{z})$ be a formula and m a natural number, then $\vdash I^{\mathrm{Diag}}_{(x),(m)}\varphi \leftrightarrow I^{\mathrm{w}}_{x,m}\varphi$. Furthermore, let $\psi(x_1, \ldots, x_n, \vec{z})$ be a formula, then $\vdash I^{\mathrm{Diag}}_{(x_i)_{i=1,\ldots,n}}\psi \leftrightarrow I^{\mathrm{Diag}}_{(x_i)_{i=1,\ldots,n},(0)_{i=1,\ldots,n}}\psi$.

Now we will define the variant of the multi-clause induction rule for which we will provide an unprovability result.

Definition 4.2.24. *The big-step multi-clause induction rule* BSMCIND *is given by*

$$\frac{\overline{L_1}(\vec{t}) \vee C_1 \quad \ldots \quad \overline{L_n}(\vec{t}) \vee C_n}{cnf\left(sk^{\exists}\left(I^{\mathrm{Diag}}_{\vec{x},\vec{m}}\bigvee_{i=1}^{n} L_i(\vec{x})\right)\right)}\ \mathrm{MCIND}_{\mathrm{S}},$$

where $L_1(\vec{x})$, …, $L_n(\vec{x})$ are literals, $\vec{t}$ is a vector of ground terms, C_1, …, C_n are clauses, and $\vec{m}$ is a vector of natural numbers.

It is trivial to give an upper bound for this rule in terms of the heterogeneous big-step diagonal induction schema given above. However, we will show that we can simplify this induction schema to obtain an upper bound that is more convenient to work with. More precisely, we will show that the induction schema that allows different step sizes for each variable is just as strong as the restriction of that schema that uses the same step size for all variables.

Definition 4.2.25. *Let $m \in \mathbb{N}$ and $\varphi(x_1, \ldots, x_n, \vec{z})$, then $I^{\mathrm{Diag}}_{\vec{x};m}\varphi \coloneqq I^{\mathrm{Diag}}_{\vec{x},\vec{m}}\varphi$, where $\vec{m} = (m)_{i=1,\ldots,n}$. Let Φ be a set of formulas, then the theory Φ-$\mathrm{BSDiag}_\omega\mathrm{IND}$ is axiomatized by the universal closure of the formulas $I^{\mathrm{Diag}}_{\vec{x};m}\varphi$ with $\varphi(\vec{x}, \vec{z}) \in \Phi$ and $m \in \mathbb{N}$.*

Lemma 4.2.26. *Let $\varphi(x_1, \ldots, x_n, \vec{z})$ be a formula with $n \geq 1$, $\vec{m} = (m_1, \ldots, m_n)$ a finite sequence of natural numbers, and $k = \min\{m_1, \ldots, m_n\}$, then*

$$\mathrm{Pred} \vdash I^{\mathrm{Diag}}_{\vec{x};k}\varphi(s^{m_1-k}(x_1), \ldots, s^{m_n-k}(x_n), \vec{z}) \to I^{\mathrm{Diag}}_{\vec{x},\vec{m}}\varphi.$$

Proof. We start by assuming

$$(\forall \vec{x}_{<i})(\forall \vec{x}_{>i})\varphi(\vec{x}_{<i}, \overline{j}, \vec{x}_{>i}, \vec{z}), \text{for } i = 1, \ldots, n \text{ and } j = 1, \ldots, m_i, \tag{$*$}$$

$$(\forall \vec{x})\Big(\varphi(\vec{x}, \vec{z}) \to \varphi(s(s^{\vec{m}}(\vec{x})), \vec{z})\Big). \tag{$\star$}$$

From $I^{\text{Diag}}_{\vec{x};k}\varphi$, $(*)$, and $(\star)$ we readily obtain

$$(\forall \vec{x})\varphi(s^{m_1-k}(x_1), \ldots, s^{m_n-k}(x_n), \vec{z}). \tag{$\dagger$}$$

Let $x_1, \ldots, x_n$ be arbitrary but fixed. We proceed by case analysis on $x_1, \ldots, x_n$. If $x_i = \overline{i}$ for some $i \in \{0, \ldots, m_i - k - 1\}$, then we are done by the $(*)$. Otherwise, $x_i = s^{m_i-k}x'_i$ for some x'_i for $i = 1, \ldots, n$. Hence, we obtain $\varphi(\vec{x})$ by $(\dagger)$. □

Thanks to the lemma above, we can now easily justify the induction schema of Definition 4.2.23 in terms of diagonal induction without big-steps (see Definition 4.2.1).

Lemma 4.2.27. *Let $\varphi(\vec{x}, \vec{z})$ be a formula and $k \in \mathbb{N}$, then $\vdash I^{\text{Diag}}_{\vec{x}} \bigwedge_{i=0}^{k} \varphi(s^i(\vec{x})) \to I^{\text{Diag}}_{\vec{x};k}\varphi$.*

Proof. Straightforward. □

Remark 4.2.28. *The reduction of big-steps to conjunction is seemingly not possible on more complicated inductive datatypes such as lists. Proving this claim is not difficult but would go beyond the scope of this section.*

We are now ready to give the upper bound for the rule BSMCIND.

Proposition 4.2.29. *Let $\mathcal{S}$ be a sound saturation system and T a Skolem-free $\exists_2$ theory. If $\mathcal{S}$ + BSMCIND refutes the clause set $cnf(sk^{\exists}(T))$, then the theory*

$$T + \text{Clause}(\mathcal{L}(T))\text{-BSDiag}_{\omega}\text{IND}$$

is inconsistent.

Proof. Proceed analogously to the proofs of Corollary 4.2.8 and Proposition 4.2.7 in order to obtain an upper bound in terms of the induction schema of Definition 4.2.23 and apply Lemma 4.2.26. □

Our method of upper bounds is in particular not able to distinguish between a method based on the induction rule BSMCIND and an analogous rule based the induction axioms $I^{\text{Diag}}_{x;k} \bigvee_{i=1}^{n} L_i$ where $L_i(\vec{x})$ is a literal and $i = 1, \ldots, n$. However, we expect that such rules may behave differently due to their analyticity. We leave a more detailed analysis of the variants of these rules as future work. Nevertheless, Lemma 4.2.26 simplifies some of the following considerations.

We will now develop a simple unprovability result for sound saturation systems extended by the rule BSMCIND.

Definition 4.2.30. *Let A and B be unary predicate symbols, then the theory $\mathcal{A}_3$ extends the theory $\mathcal{A}_0$ by the universal closure of the following formulas*

$$A(0), \tag{A3.1}$$
$$B(0), \tag{A3.2}$$
$$A(x) \wedge B(x) \to A(s(x)) \wedge B(s(x)). \tag{A3.3}$$

We now construct a suitable model of $\mathcal{A}_3 + \mathcal{L}(\mathcal{A}_3)$-BSDiag$_\omega$IND.

Definition 4.2.31. *Let $n \in \mathbb{N}$, then the n-th triangular number is given by*

$$\triangle_n = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}.$$

Definition 4.2.32. *Let M_3 be the $\mathcal{L}(\mathcal{A}_3)$ structure consisting of pairs of the form (i, n) where $i \in \{0, 1\}$ and $n \in \mathbb{Z}$ such that $i = 0$ implies $n \in \mathbb{N}$. Furthermore, let M_3 interpret the non-logical symbols as follows*

$$0^{M_3} = (0, 0),$$
$$s^{M_3}((i, n)) = (i, n + 1),$$
$$A^{M_3} = (\{0\} \times \mathbb{N}) \cup \left(\{1\} \times \bigcup_{\substack{n \in \mathbb{N} \\ n \text{ is even}}} \{-\triangle_{n+1} + 1, \ldots, -\triangle_n\} \right),$$
$$B^{M_3} = (\{0\} \times \mathbb{N}) \cup \left(\{1\} \times \bigcup_{\substack{n \in \mathbb{N} \\ n \text{ is odd}}} \{-\triangle_{n+1} + 1, \ldots, -\triangle_n\} \right).$$

Lemma 4.2.33. $M_3 \models \mathcal{A}_3 + \text{Pred} + \text{ACY}$.

Proof. Observe that $M_3|\mathcal{L}_0 = M_2|\mathcal{L}_0$, hence $M_3 \models \mathcal{A}_0 + \text{Pred} + \text{ACY}$. Furthermore, it is clear that $M_3 \models A(0) \wedge B(0)$. For axiom (A3.3) let (i, n) be an element of M_3. For the case where $i = 0$, it suffices to observe that $(i, n + 1) \in A^{M_3} \cap B^{M_3}$. For the case $i = 1$ observe that either $(i, n) \notin A^{M_3}$ or $(i, n) \notin B^{M_3}$. □

We are now ready to show that M_3 is a model of big-step diagonal induction over clauses.

Proposition 4.2.34. $M_3 \models \text{Clause}(\mathcal{L}(\mathcal{A}_3))$-BSDiag$_\omega$IND.

Proof. Let $m \in \mathbb{N}$, $C(\vec{x}, \vec{z})$ a $\mathcal{L}(\mathcal{A}_3)$ clause, and $\vec{c}$ a finite sequence of elements of M_3 such

that $|\vec{z}| = |\vec{c}|$. Now assume that

$$M_3 \models \bigwedge_{i=1}^{|\vec{x}|} \bigwedge_{j=0}^{m} C(\vec{x}_{<i}, \overline{j}, \vec{x}_{>i}, \vec{c}), \tag{$*$}$$

$$M_3 \models C(\vec{x}, \vec{c}) \to C(s^{m+1}(\vec{x}), \vec{c}). \tag{$\star$}$$

By Lemma 4.2.33, $M_3 \models \mathcal{A}_0 + \text{Pred} + \text{ACY}$. Thus by Corollary 2.6.4, we have $M_3 \models \mathcal{L}_0$-IND. Hence, by Lemmas 4.2.3 and 4.2.27, we have $M_3 \models \text{Clause}(\mathcal{L}_0)\text{-BSDiag}_\omega\text{IND}$. Therefore, we can assume without loss of generality that C contains a non-equational literal containing an occurrence of an induction variable. Let $\vec{d} = (d_1, \ldots, d_{|\vec{x}|})$ be a vector of elements of $\mathcal{D}(M_3)$ with $d_i = (b_i, n_i)$ for $i = 1, \ldots, |\vec{x}|$ and $I := \{i \in \{1, \ldots, |\vec{x}|\} \mid b_i = 0\}$. Let us start by considering the case where $I \neq \varnothing$. For each $i \in I$ let $k_i := n_i \bmod (m+1)$. Let $i_0 \in I$ be such that k_{i_0} is least among the k_i with $i \in I$. Therefore, we have $d'_i = (b_i, n_i - k_{i_0}(m+1)) \in \mathcal{D}(M_3)$ for $i = 1, \ldots, |\vec{x}|$. Moreover, by $(*)$ we have $M_3 \models C(\vec{d'}, \vec{c})$. By k_{i_0}-fold application of $(\star)$ we obtain $M_3 \models C(\vec{d}, \vec{c})$.

Now we consider the case where $I = \varnothing$. We will only consider the case where C contains a literal of the form $A(s^k(x_i))$. The cases where C contains a literal of the form $\neg A(s^k(x_i))$, $B(s^k(x_i))$, or $\neg B(s^k(x_i))$ are analogous. By the construction of M_3 there exists a natural number l such that $-\triangle_l \leq n_i + k$ and $|\triangle_l - (\triangle_{l+1} - 1)| \geq m+1$. Hence there exists $\triangle_l \leq p < \triangle_{l+1}$ and $q \in \mathbb{N}$ such that $-p + q(m+1) = n_i + k$. By the construction of M_3 we have $(1, -p) \in A^{M_3}$. Hence, $M_3 \models A(s^k((1, -p-k)))$, that is,

$$M_3 \models C(d'_1, \ldots, d'_{i-1}, (1, -p), d'_{i+1}, \ldots, d'_{|\vec{x}|}, \vec{c}),$$

where $d'_i = (b_i, n_i - q(m+1))$ for $i = 1, \ldots, i-1, i+1, \ldots, |\vec{x}|$. Hence, by a q-fold application of $(\star)$ we now obtain $M_3 \models C(\vec{d}, \vec{c})$. □

Proposition 4.2.35. $\mathcal{A}_3 + \text{Clause}(\mathcal{L}(\mathcal{A}_3))\text{-BSDiag}_\omega\text{IND} \nvdash A(x)$.

Proof. By Proposition 4.2.34 it suffices to observe that $(1,1) \notin A^{M_3}$. □

This gives us an unprovability result for the big-step multi-clause induction rule.

Theorem 4.2.36. *Let $\mathcal{S}$ be a sound and saturation system, then $\mathcal{S}$ + BSMCIND does not refute the clause set $cnf(sk^{\exists}(\mathcal{A}_3 + \exists x \neg A(x)))$.*

Proof. Proceed indirectly and assume that $\mathcal{S}$+BSMCIND refutes $cnf(sk^{\exists}(\mathcal{A}_3 + \exists x \neg A(x)))$, then by Proposition 4.2.29 the theory $\mathcal{A}_3 + \exists x \neg A(x) + \text{Clause}(\mathcal{L}(\mathcal{A}_3))\text{-BSDiag}_\omega\text{IND}$ is inconsistent. This contradicts Proposition 4.2.35. □

In order to overcome this unprovability a system has to implement an induction mechanism that provides "more conjunction" than provided by big-step induction. The following is an example of a natural extension of the multi-clause induction rule that overcomes

the unprovability of Proposition 4.2.35:

$$\frac{D_1(\vec{t}) \vee C_1 \quad \dots \quad D_n(\vec{t}) \vee C_n}{cnf(sk^{\exists}(I_{\vec{x}} \bigvee_{i=1}^{n} \overline{D_n}))}$$

where $D_1(\vec{x}), \dots, D_n(\vec{x})$ are clauses and $\vec{t}$ is a finite sequence of terms of terms with $|\vec{x}| = |\vec{t}|$. Of course, implementing such a rule efficiently is difficult because of the many possible ways in which the rule can be applied and each application generates even more clauses.

We could go on and investigate stronger variants of big-step induction, however, this would digress to far from the purpose of this section. Instead we conclude this section with a conjecture that covers many of the variants relevant in practice.

Definition 4.2.37. *Let $\varphi(x, \vec{z})$ be a formula and m a natural number, then the strong big-step induction axiom $I^{\mathrm{S}}_{x,m}\varphi$ is given by*

$$\bigwedge_{k=0}^{m} \varphi(\overline{k}, \vec{z}) \wedge (\forall x) \left(\bigwedge_{j=0}^{m} \varphi(s^j(x), \vec{z}) \rightarrow \varphi(s^{m+1}(x), \vec{z}) \right) \rightarrow (\forall x)\varphi(x, \vec{z}).$$

Let Φ be a set of formulas, then Φ-$\mathrm{BS_S IND}$ is axiomatized by the universal closure of the formulas $I^{\mathrm{S}}_{x,m}\varphi$, where $\varphi \in \Phi$ and $m \geq 1$ is a natural number.

Conjecture 4.2.38. $\mathcal{A}_3 + \bigcup_{k\in\mathbb{N}} \forall_k \mathrm{Clause}(\mathcal{L}(\mathcal{A}_3))\text{-}\mathrm{BS_S IND} \nvdash A(x)$.

4.3 Towards analytic unprovability results

In the previous sections we have provided a variety of unprovability results for sound saturation systems extended by variants of the single-clause or the multi-clause induction rules. We have established these results by providing upper bounds on the strength of the systems in terms of logical theories. This essentially abstracts all the details of the underlying calculi of the systems and allows us to work model-theoretically. On the other hand, this technique is not sensitive to the analytic nature of certain induction mechanisms. That is, our technique can not provide examples where a system fails because the underlying calculus does not generate enough syntactic material. By taking into account analyticity restrictions we may be able to find much more elementary and thus more practically relevant unprovability results.

In this section we demonstrate a simple preliminary unprovability result for a variant of the single-clause induction rule considered in Section 4.1. The result that we provide improves upon the results of Section 4.1 in the sense that it shows that the rigid single-clause induction can not deal with some properties that can be proven by a simple argument by induction with induction parameters. In particular this complements the empirical observations of [Haj+20] that justify the necessity of considering more powerful rules such as the single-clause induction rule described in [Haj+20].

We will consider the extension of a variant of the paramodulation calculus by a rigid single-clause induction rule. We start by recalling the rules of the paramodulation calculus, which is a calculus that underlies many of the modern saturation provers and is thus a suitable abstraction.

Definition 4.3.1. *The resolution rule, factoring rule, exchange rule, paramodulation rule, and the reflexivity rule are given by*

$$\frac{C \vee L_1 \qquad D \vee L_2}{(C \vee D)\sigma}\ \text{Res}, \quad \frac{C \vee L_1 \vee L_2}{C\sigma}\ \text{Fac}, \quad \frac{C \vee u \neq v}{C\sigma}\ \text{Refl},$$
$$\frac{C \vee L_1 \vee L_2 \vee D}{C \vee L_2 \vee L_1 \vee D}\ \text{Ex} \quad \frac{C \vee u = v \qquad D \vee L(v', \vec{z})}{(C \vee D \vee L(u, \vec{z}))\sigma}\ \text{Para},$$

where for the rule Res, *σ is the most general unifier of L_1 and $\overline{L_2}$, for the rule* Fac, *σ is the most general unifier of L_1 and L_2, for the rule* Refl, *σ is the most general unifier of u and v, and for the rule* Para, *σ is the most general unifier of v and v'. The resolution calculus is denoted by $\mathcal{R}$ and is the saturation system consisting of the rules* Res, Fac, *and* Ex. *The paramodulation calculus $\mathcal{P}$ is the system consisting of the rules of $\mathcal{R}$ and the rules* Refl *and* Para.

Remark 4.3.2 (Exchange rule)**.** *Since we treat clauses as formulas the rules above are sensitive to the position of the literals. We have added the exchange rule to overcome this limitation.*

In the following we will provide an unprovability result for an extension of the resolution calculus by a variant of the rigid single-clause induction rule considered in Section 4.1, whose instances are of the form

$$\frac{\overline{L}(a) \vee C}{\neg L(0) \vee L(c) \vee L(x), \neg L(0) \vee \neg L(s(c)) \vee L(x)}\ \text{RSCIND}'_{\text{S}},$$

where a is nullary function symbol, $L(x)$ is a literal free of a, and

$$sk^{\forall}((\forall x)(L(x) \to L(s(x)))) = L(c) \to L(s(c)).$$

The clause set that we show unprovable in the resolution calculus extended by $\text{RSCIND}'_{\text{S}}$ is very similar to the one of Definition 6.4.14 and the unprovability relies on a lack of induction parameters. We will denote by I the set consisting of the sentences

$$I_x A(s^{k_1}(x), s^{k_2}(x)) \text{ and } I_x \neg A(s^{k_1}(x), s^{k_2}(x))$$

with $k_1, k_2 \in \mathbb{N}$.

Proposition 4.3.3. *Let $A/2$ be a predicate symbol, then the system $\mathcal{R} + \text{RSCIND}'_{\text{S}}$ does*

not refute the clause set consisting of the clauses

$$A(0, y), \tag{C1}$$
$$\neg A(x, y) \vee A(s(x), y), \tag{C2}$$
$$\neg A(c, c), \tag{C3}$$

where $sk^{\forall}((\forall x)A(x, x)) = A(c, c)$.

Proof. Let $\mathcal{A}'_5 \coloneqq \mathcal{A}_5 + \text{C3}$, $\mathcal{L} = \{0, s, A, c\}$, and $\mathcal{C}_0, \ldots, \mathcal{C}_n$ a $\mathcal{R} + \text{RSCIND}'_{\text{S}}$ deduction with $\mathcal{C}_0 = \{\text{C1}, \text{C2}, \text{C3}\}$. We proceed by induction on n to show

$$\mathcal{L}(\mathcal{C}_n) \subseteq \{c\} \cup \mathcal{L}(sk^{\exists}(I))),$$
$$\mathcal{A}'_5 + sk^{\exists}(I) \vdash \mathcal{C}_n,$$

and that moreover $\mathcal{C}_n$ can be partitioned into clause sets $\mathcal{C}_n^{(1)}$, $\mathcal{C}_n^{(2)}$, $\mathcal{C}_n^{(3)}$, and $\mathcal{C}_n^{(4)}$ such that

$$\mathcal{C}_n^{(1)} = \{\neg A(c, c)\},$$
$$\mathcal{C}_n^{(2)} \subseteq \{A(\overline{k}, y) \mid k \in \mathbb{N}\},$$
$$\mathcal{C}_n^{(3)} \subseteq \{\neg A(x, y) \vee A(s^k x, y), A(s^k x, y) \vee \neg A(x, y) \mid k \in \mathbb{N}, k \geq 1\},$$

and $\mathcal{C}_n^{(4)}$ is a set of clauses such that $c \notin \mathcal{L}(\mathcal{C}_n^{(4)})$ whose literals are pairwise variable disjoint and whose atoms are of the form

$$A(s^{k_1}(\nu), s^{k_2}(\nu)),$$

where $k_1, k_2 \in \mathbb{N}$ and ν is either a nullary function symbol or a variable. For the base case $n = 0$ we have $\mathcal{L}(\mathcal{C}_0) = \{0, s, A, c\}$ and $\mathcal{A}'_5 \vdash \mathcal{C}_0$. Furthermore, we let $\mathcal{C}_0^{(1)} = \{\text{C3}\}$, $\mathcal{C}_0^{(2)} = \{\text{C1}\}$, $\mathcal{C}_0^{(3)} = \{\text{C2}\}$, and $\mathcal{C}_0^{(4)} = \varnothing$. For the induction step we consider the clause set $\mathcal{C}_{n+1}$. We proceed by case distinction on the inference by which $\mathcal{C}_{n+1}$ was obtained.

Assume that $\mathcal{C}_{n+1}$ is obtained by a resolution inference, then there are clauses $C, D \in \mathcal{C}_n$ such that $\mathcal{C}_{n+1} = \mathcal{C}_n \cup \{R\}$, where R is the resolvent of C and D. Then, clearly $\mathcal{A}'_0 + sk^{\exists}(I) \vdash \mathcal{C}_{n+1}$ and the language is not extended. Hence it remains to analyze the shape of the resulting clause:

- Assume that $C \in \mathcal{C}_n^{(1)}$, then $C = \neg A(c, c)$ and cannot resolve with any clause in $\mathcal{C}_n^{(1)}$, $\mathcal{C}_n^{(2)}$, or $\mathcal{C}_n^{(3)}$. Hence $D \in \mathcal{C}_n^{(4)}$ and D is of the form $D'(\vec{y}) \vee A(x, x)$. Thus, $R = D'$ we let $\mathcal{C}_{n+1}^{(4)} = \mathcal{C}_n^{(4)} \cup \{D'\}$. Clearly $\mathcal{C}_{n+1}^{(4)}$ satisfies the invariant.

- Assume that $C \in \mathcal{C}_n^{(2)}$, then $C = A(\overline{k}, y)$ for some $k \in \mathbb{N}$ and C can resolve with clauses from $\mathcal{C}_n^{(3)}$ or $\mathcal{C}_n^{(4)}$. If $D \in \mathcal{C}_n^{(3)}$, then D is of the form $A(s^{k'}x, y) \vee \neg A(x, y)$ for some $k' \geq 1$ and we have $R = A(\overline{k + k'}, y)$. Thus $\mathcal{C}_{n+1}^{(2)} = \mathcal{C}_n^{(2)} \cup \{R\}$ and we are done. If $D \in \mathcal{C}_n^{(4)}$, then $D = D'(\vec{y}) \vee \neg A(s^{k_1}(\nu), s^{k_2}(\nu))$, where ν is either a nullary

function symbol or a variable. Since the literals of D are pairwise variable disjoint, we have in both cases $R = D'$. Thus we let $\mathcal{C}_{n+1}^{(4)} = \mathcal{C}_n^{(4)} \cup \{R\}$ and we are done.

- Assume that $C \in \mathcal{C}_n^{(3)}$ and that C is of the form $\neg A(x,y) \vee A(s^k(x),y)$ with $k \geq 1$. Then C can neither resolve with the clause $\neg A(c,c)$ nor with the clauses in $\mathcal{C}_n^{(2)}$. If $D \in \mathcal{C}_n^{(3)}$, then D is of the form $A(s^{k'}(x),y) \vee \neg A(x,y)$. Hence, R is of the form $\neg A(x,y) \vee A(s^{k+k'}(x),y)$, that is, we let $\mathcal{C}_{n+1}^{(3)} \cup \{R\}$. If D is in $\mathcal{C}_n^{(4)}$, then D is of the form $D'(\vec{y}) \vee \neg A(s^{k_1}(\nu), s^{k_2}(\nu))$. If ν is a nullary function symbol, then $k_1 \geq k$ and $R = \neg A(s^{k_1-k}(\nu), s^{k_2}(\nu) \vee D'$. Hence, we let $\mathcal{C}_{n+1}^{(4)} = \mathcal{C}_n^{(4)} \cup \{R\}$. Now let us consider the case ν is a variable. If $k \leq k_1$, then $R = \neg A(s^{k_1-k}(\nu), s^{k_2}(\nu)) \vee D'$ and we let $\mathcal{C}_{n+1}^{(4)} = \mathcal{C}_n^{(4)} \cup \{R\}$. If $k > k_1$, then $R = \neg A(\nu, s^{k_2+(k-k_1)}(\nu)) \vee D'$ and we let again $\mathcal{C}_{n+1}^{(4)} = \mathcal{C}_n^{(4)} \cup \{R\}$.

  Now assume that C is of the form $A(s^k(x),y) \vee \neg A(x,y)$. If $D \in \mathcal{C}_n^{(2)}$, then D is of the form $A(\overline{k'},y)$ with $k' \in \mathbb{N}$. Then $R = A(\overline{k+k'},y)$ and we let $\mathcal{C}_{n+1}^{(2)} = \mathcal{C}_n^{(2)} \cup \{R\}$.

  The cases where $D \in \mathcal{C}_n^{(3)}$ or $D \in \mathcal{C}_n^{(4)}$ are analogous to the cases above.

- Assume that $C \in \mathcal{C}_n^{(4)}$. The cases where $D \in \mathcal{C}_n^{(i)}$ for $i \in \{1,2,3\}$ are symmetric to the cases considered above. Hence, we only need to consider the case with $D \in \mathcal{C}_n^{(4)}$. Assume that $C = C'(\vec{z}_1) \vee L$, $D = D'(\vec{z}_2) \vee M$, and $R = C'\sigma \vee D'\sigma$, where $L = A(s^{k_1}(\nu_1), s^{k_2}(\nu_1))$, $M = \neg A(s^{m_1}(\nu_2), s^{m_2}(\nu_2))$, ν_i is either a constant or a variable not in $\vec{z}_i$ for $i = 1,2$, and σ is a most general unifier of L and $\overline{M}$. The variables appearing in the domain of the unifier σ are a subset of $\{\nu_1, \nu_2\}$, hence we have $R = C' \vee D'$. The case where the polarities of L and M are inverted is symmetric, hence $\mathcal{C}_{n+1}^{(4)} = \mathcal{C}_n^{(4)} \cup \{R\}$. The case where $C = C'(\vec{z_1}) \vee M$ and $D = D'(\vec{z_2}) \vee L$ is analogous.

Now assume that $\mathcal{C}_{n+1}$ is obtained by an application of the factor rule. Then $\mathcal{C}_{n+1} = \mathcal{C}_n \cup \{F\}$, where F is a factor of C and $C \in \mathcal{C}_n$. Clearly, $\mathcal{L}(\mathcal{C}_{n+1}) \subseteq \mathcal{L}(\mathcal{C}_n)$ and we have $\mathcal{A}_5' + I \vdash \mathcal{C}_{n+1}$. Hence it remains to analyze the shape of the clause F. Clearly, $C \notin \mathcal{C}_n^{(1)}, \mathcal{C}_n^{(2)}, \mathcal{C}_n^{(3)}$, that is, $C \in \mathcal{C}_n^{(4)}$. We have $C = C'(\vec{z}) \vee L \vee M$ and $F = C' \vee L\sigma$ where $L = P(s^{k_1}(\nu_1), s^{k_2}(\nu_1))$, $M = P(s^{m_1}(\nu_2), s^{m_2}(\nu_2))$, $P \in \{A, \overline{A}\}$, σ is a most general unifier of L and M, and ν_i is either a constant symbol or a variable not in $\vec{z}$ for $i = 1,2$. Clearly, $L\sigma = M\sigma = P(s^{n_1}(\nu'), s^{n_2}(\nu'))$ for some $n_1, n_2 \in \mathbb{N}$ and some $\nu' \in \{\nu_1, \nu_2\}$. Hence, we let $\mathcal{C}_{n+1}^{(4)} = \mathcal{C}_n^{(4)} \cup \{F\}$.

Assume that $\mathcal{C}_{n+1}$ is obtained by an application of the exchange rule. Then there is a clause $C = C_1 \vee L_1 \vee L_2 \vee C_2 \in \mathcal{C}_n$ such that $\mathcal{C}_{n+1} = \mathcal{C}_n \cup \{E\}$ where $E = C_1 \vee L_2 \vee L_1 \vee C_2$. Hence, either $C \in \mathcal{C}_n^{(3)}$ or $C \in \mathcal{C}_n^{(4)}$. If $C \in \mathcal{C}_n^{(3)}$, then we let $\mathcal{C}_{n+1}^{(3)} = \mathcal{C}_n^{(3)} \cup \{E\}$. Otherwise, we let $\mathcal{C}_{n+1}^{(4)} = \mathcal{C}_n^{(4)} \cup \{E\}$.

Assume that $\mathcal{C}_{n+1}$ is obtained from $\mathcal{C}_n$ by an application of the rule $\mathrm{RSCIND}_{\mathrm{S}}'$, then there is $C \vee P(s^{k_1}d, s^{k_2}d) \in \mathcal{C}_n$, where d is a constant symbol and $P \in \{A, \overline{A}\}$. Moreover,

$\mathcal{C}_{n+1}$ consists of the clauses of $\mathcal{C}_n$ and the following clauses

$$P(s^{k_1}0, s^{k_2}0) \vee \overline{P}(s^{k_1}d', s^{k_2}d') \vee \overline{P}(s^{k_1}x, s^{k_2}x), \tag{D_1}$$
$$P(s^{k_1}0, s^{k_2}0) \vee P(s^{k_1+1}d', s^{k_2+1}d') \vee \overline{P}(s^{k_1}x, s^{k_2}x), \tag{D_2}$$

where $d' = \mathfrak{s}_{(\forall x)(P(s^{k_1}(x),s^{k_2}(x))\to P(s^{k_1}(x),s^{k_2}(x)))}$. We let $\mathcal{C}^{(4)}_{n+1} = \mathcal{C}_n \cup \{D_1, D_2\}$. Moreover, we clearly have $sk^{\exists}(I_x P(s^{k_1}x, s^{k_2}x)) \models \{D_1, D_2\}$ and therefore $\mathcal{A}'_5 + sk^{\exists}(I) \models \mathcal{C}_{n+1}$. Furthermore, $\mathcal{L}(\mathcal{C}_{n+1}) \subseteq \{c\} \cup \mathcal{L}(sk^{\exists}(I))$ and the choice of $\mathcal{C}^{(4)}_{n+1}$ satisfies the invariant.

Now assume that $\mathcal{C}_n$ contains the empty clause. Then by the above the theory $\mathcal{A}'_5 + sk^{\exists}(I)$ is inconsistent. Thus also the theory $\mathcal{A}_5 + I$ is inconsistent. However, this contradicts Corollary 6.4.15. □

On the other hand, if in the clause set of Proposition 4.3.3 one of the occurrences of c is replaced by another constant symbol, then the system $\mathcal{R} + \mathrm{RSCIND}'_{\mathrm{S}}$ refutes the resulting clause set.

Remark 4.3.4. *Proposition 4.3.3 also holds if Clause* C3 *is replaced by the clause* $\neg A(c, c)$ *where* $c/0$ *is a function symbol such that* $c \notin \mathcal{L}(I)$.

Interestingly, as we will later show in Chapter 6, the clause set of Proposition 4.3.3 is essentially also not refuted by induction systems based on clause set cycles. However, in the case of clause set cycles we will not even exploit the underlying calculus.

The result above suggests that the rigid single-clause induction rule may have problems to deal with induction parameters. However, the following lemma together with Lemma 3.2.4 shows that the ability of the system $\mathcal{P} + \mathrm{RSCIND}'_{\mathrm{S}}$ to handle induction parameters depends on the language of the input clause set.

Lemma 4.3.5. *Let* $c/0$ *be a function symbol, then the system* $\mathcal{P} + \mathrm{RSCIND}'_{\mathrm{S}}$ *refutes the clause set consisting of the following clauses*

$$x + 0 = x,$$
$$x + s(y) = s(x + y),$$
$$c + c = c,$$
$$c \neq 0.$$

Proof. We start by applying single-clause induction to the clause $c \neq 0$ on the constant to obtain in particular the clause

$$0 \neq 0 \vee c_1 = 0 \vee x = 0. \tag{4.1}$$

We apply paramodulation from $x + 0 = x$ into $c \neq 0$ in order to obtain the clause

$$c + 0 \neq 0 + 0. \tag{4.2}$$

Now we apply paramodulation from (4.1) into (4.2) in order to obtain the clause

$$0 \neq 0 \vee x = 0 \vee c + c_1 \neq 0 + c_1. \tag{4.3}$$

Thus, we can apply the induction rule $\mathrm{RSCIND}'_{\mathrm{S}}$ to the literal $c + c_1 \neq 0 + c_1$ on constant c_1 in order to obtain the clause

$$c + 0 \neq 0 + 0 \vee c + c_2 = 0 + c_2 \vee c + x = 0 + x. \tag{4.4}$$

Now we apply the induction rule $\mathrm{RSCIND}'_{\mathrm{S}}$ to the literal $c + c_2 = 0 + c_2$ on the constant c_2 in order to obtain the clauses

$$c + 0 = 0 + 0 \vee c + c_3 \neq 0 + c_3 \vee c + x \neq 0 + x, \tag{4.5}$$
$$c + 0 = 0 + 0 \vee c + s(c_3) = 0 + s(c_3) \vee c + x \neq 0 + x. \tag{4.6}$$

First we use paramodulation from $x + 0$ to (4.5) and (4.6), $x + s(y) = s(x + y)$, and resolve with $c \neq 0$ to obtain the clauses

$$c + c_3 \neq 0 + c_3 \vee c + x \neq 0 + x, \tag{4.7}$$
$$s(c + c_3) = s(0 + c_3) \vee c + x \neq 0 + x. \tag{4.8}$$

Now we use paramodulation from (4.7) into (4.8) followed an application of the reflexivity and the factor rule in to obtain the clause

$$c + x \neq 0 + x. \tag{4.9}$$

We apply paramodulation from the clause $c + c = c$ into (4.9) to obtain

$$c \neq 0 + c. \tag{4.10}$$

Hence, we can apply the induction rule $\mathrm{RSCIND}'_{\mathrm{S}}$ in order to obtain the clauses

$$0 \neq 0 + 0 \vee c_4 = 0 + c_4 \vee x = 0 + x, \tag{4.11}$$
$$0 \neq 0 + 0 \vee s(c_4) = 0 + s(c_4) \vee x = 0 + x. \tag{4.12}$$

We apply paramodulation from $x + 0$ and $x + s(y) = s(x + y)$ to (4.13) and (4.14) in order to obtain the clauses

$$c_4 = 0 + c_4 \vee x = 0 + x, \tag{4.13}$$
$$s(c_4) = s(0 + c_4) \vee x = 0 + x. \tag{4.14}$$

Now we use paramodulation from (4.13) into (4.14) followed by an application of reflexivity and factoring to obtain the clause

$$x = 0 + x.$$

Finally, use paramodulation from $x = 0 + x$ into (4.10) to obtain $c \neq c$ and apply the reflexivity rule to obtain the empty clause. □

Furthermore, the rule $\mathrm{RSCIND}'_{\mathrm{S}}$ is also powerful enough to prove the univariate instance of associativity of the addition on natural numbers from the usual primitive recursive defining equations of addition.

Lemma 4.3.6. *Let $c/0$ be a function symbol, then the system $\mathcal{P} + \mathrm{RSCIND}'_{\mathrm{S}}$ refutes the clause set consisting of the following clauses*

$$x + 0 = x, \tag{4.15}$$
$$x + s(y) = s(x + y), \tag{4.16}$$
$$c + (c + c) \neq (c + c) + c. \tag{4.17}$$

Proof. Let us define $A(x, y, z) \coloneqq x + (y + z) = (x + y) + z$. In the initial clauses we have the clause $\overline{A}(c, c, c)$. Hence, the rule $\mathrm{RSCIND}'_{\mathrm{S}}$ applied to $A(c, c, c)$ on the term c gives the clauses

$$0 + (0 + 0) \neq (0 + 0) + 0 \vee A(c_1, c_1, c_1) \vee A(x, x, x) \tag{4.18}$$
$$0 + (0 + 0) \neq (0 + 0) + 0 \vee \overline{A}(s(c_1), s(c_1), s(c_1)) \vee A(x, x, x). \tag{4.19}$$

Apply paramodulation from (4.15) to (4.18) in order to obtain the clause

$$\underbrace{0 + 0 \neq 0}_{\coloneqq \overline{B}(0)} \vee A(c_1, c_1, c_1) \vee A(x, x, x), \tag{4.20}$$

where $B(x) \equiv x + x = x$. Now we use $\mathrm{RSCIND}'_{\mathrm{S}}$ on the literal $\overline{B}(0)$ and the constant 0 in order to obtain among others the clause

$$\overline{B}(0) \vee B(c_2) \vee B(x). \tag{4.21}$$

We apply paramodulation from (4.21) into $\overline{A}(c, c, c)$ followed by factorization in order to obtain

$$\overline{B}(0) \vee B(c_2) \vee c + c \neq c + c. \tag{4.22}$$

Now apply paramodulation from (4.15) into (4.23) in order to obtain the clause

$$\overline{B}(0) \vee B(c_2) \vee \underbrace{c + (c + 0) \neq (c + c) + 0}_{= \overline{A}(c,c,0)}. \tag{4.23}$$

We can apply $\mathrm{RSCIND}'_{\mathrm{S}}$ on the literal $\overline{A}(c, c, 0)$ and the constant 0 in order to obtain the clauses

$$\overline{A}(c, c, 0) \vee A(c, c, c_3) \vee A(c, c, x), \tag{4.24}$$
$$\overline{A}(c, c, 0) \vee \overline{A}(c, c, s(c_3)) \vee A(c, c, x). \tag{4.25}$$

After that, we use paramodulation from (4.15) into (4.24) and (4.25) followed by reflexivity to obtain the clauses

$$A(c, c, c_3) \vee A(c, c, x), \tag{4.26}$$

$$\overline{A}(c, c, s(c_3)) \vee A(c, c, x). \tag{4.27}$$

Since $A(c, c, s(c_3)) \equiv c + (c + s(c_3)) = (c + c) + s(c_3)$ we can by paramodulation from (4.16) into (4.27) obtain the clause

$$s(c + (c + c_3)) \not\approx s((c + c) + c_3) \vee A(c, c, x). \tag{4.28}$$

Now use paramodulation from (4.26) into (4.28) followed by factorization and reflexivity to obtain

$$A(c, c, x). \tag{4.29}$$

Finally, resolve (4.29) with $\overline{A}(c, c, c)$ to obtain the empty clause. □

In the proofs above it is interesting to observe that we have used the induction rule $\mathrm{RSCIND}'_{\mathrm{S}}$ not only for arguments by induction but also in order to introduce new syntactic material that allows us to obtain literals that are suitable for induction. In particular in the proof of Lemma 4.3.6 we have used induction on the formula $x = x + x$ which together with $x + 0 = x$ allows us to rewrite the literal $c + (c + c) \not\approx (c + c) + c$ into $c + (c + 0) \not\approx (c + c) + 0$ and only the last application of the induction rule carries out an actual argument by induction. It would be interesting to clarify to which extend this phenomenon occurs in practice because the use of induction for the introduction of new material is conflicting with the usual practice of implementations of resolution-like calculi to prune the search space. The single-clause induction rule introduced in [Haj+20] (see also Definition 4.1.3) improves this situation in two ways: Not only is the rule stronger but the rule is now also able to attempt a suitable induction more directly and may thus possibly result in a better practical performance.

Remark 4.3.7. *The proofs of Lemmas 4.3.5 and 4.3.6 strongly rely on the variant of the paramodulation calculus that is used. For example, it is not obvious whether we could obtain a clause like Clause* (4.3) *if paramodulation is applied to all occurrences of a term, as is customary in implementations (see [Wei01]).*

In [Haj+20] it is empirically observed that the practical system considered in [RV19] based on the following variant of the rigid single-clause induction rule fails to refute the clause set in the statement of Proposition 4.3.3.

Definition 4.3.8. *The resolving rigid single-clause induction rule* ResRSCIND *consists of the instances of the form*

$$\frac{\overline{L}(a) \vee C}{\neg L(0) \vee L(c) \vee C, \neg L(0) \vee \neg L(s(c)) \vee C}\ \mathrm{ResRSCIND},$$

where $a/0$ *is a function symbol,* $L(x)$ *is a literal free of* a*, and*

$$sk^{\forall}((\forall x)(L(x) \to L(s(x)))) = L(c) \to L(s(c)).$$

The lemma above suggests that either the resolving variant of the rigid single-clause induction rule is weaker than $\mathrm{RSCIND}_{\mathrm{S}}$ or other details of the concrete system such as the search strategy or redundancy mechanisms interfere with the induction rule. It would be interesting to investigate the situation in more detail and in particular it is certainly interesting to consider the following question.

Question 4.3.9. *Does the system* $\mathcal{P} + \mathrm{ResRSCIND}$ *refute the clause set consisting of the clauses* (4.15)*,* (4.16)*, and* (4.17)*?*

4.4 Summary

In this chapter we have considered instances of the single-clause induction rule described in [RV19; Haj+20] and the multi-clause induction rule described in [Haj21; Haj+21b]. By establishing upper bounds for sound saturation systems extended by these rules we were able to give several elementary unprovability results, see Theorem 4.1.14, Corollary 4.1.24, Theorem 4.2.16, Theorem 4.2.36.

Our method of providing a logical theory that acts as an upper bound on the strength of a system under consideration has the advantage that once the formal machinery is set up, it becomes straightforward to provide unprovability results. This is partly due to the fact that we can work with semantic arguments as opposed to syntactical arguments that exploit features of the underlying calculi. Therefore, our results are challenging for AITP systems in the sense that they correspond to separations between theories and therefore require substantial extensions to an induction mechanism in order to be overcome. Furthermore, our unprovability results express basic properties such as for example that every number is even or odd, or the cancellation of a multiplicative constant. This is in contrast to other unprovability results following from Gödel's incompleteness theorems that usually express much more complicated properties such as the consistency of $I\Sigma_n$. In this sense we consider our unprovability results to be of practical relevance.

On the other hand our method is too coarse to differentiate between some variants of induction rules. For example the induction rules considered in this chapter are implemented in such a way that they immediately resolve the clauses corresponding to the induction axiom with the clauses to which the induction rules are applied. We suspect a sound saturation system extended by such a variant is strictly weaker, than the same system extended by the rule that does not resolve the clauses immediately. Our method produces the same upper bound for both variants and is therefore too weak to separate both systems. In Section 4.3 we have provided a preliminary unprovability that takes into account the underlying proof system and improves significantly on the results obtained in Section 4.1. However, providing results that exploit more fine grained analyticity properties may be very laborious and especially so in the case of statements involving equality.

Another shortcoming of the unprovability results in this section is that they do not account for mechanisms that extend the working language of a saturation system by predicate symbols. For instance, the results above do not deal with clause splitting mechanisms such as AVATAR [Vor14] that introduce predicate symbols. This particular issue will be addressed in the next chapter where a method involving such a clause splitting mechanism is considered. The results show that clause splitting does not affect the power of the induction mechanism. Another issue that we did not address in this section are the introduction of definitions by definitional translations. In this case the situation is different because such definitions may allow us to draw complex formulas into an induction rule that otherwise handles a simpler class of formulas only. However, we expect that such translations play only a small role for the simple clause sets considered in this chapter. This issue will not be addressed in this thesis and is left as future work.

5 Case study: Zipperposition

In this chapter we consider the AITP method described by Cruanes in [Cru17]. The method is similar to the methods considered in Chapter 4 in that it essentially extends an saturation system by an induction rule. However, it differs from these methods in that the induction is slightly more complex, the method provides a lemma rule, and moreover the method integrates the induction more explicitly with the splitting mechanism. Implementations of the systems considered in Chapter 4 also make use of clause splitting but this mechanism is provided transparently by the underlying saturation system.

In Section 5.1 we introduce the calculus that we will analyze in this chapter. In Section 5.2 we then provide an upper bound for this calculus in terms of a logical theory with induction and Skolem axioms. After that, we provide in Section 5.3 a completeness result in the $\exists_2$ setting. In particular this completeness result shows that the calculus considered in this chapter is at least as strong as the systems considered in Chapter 4. Finally, in Section 5.4 we make use of the upper bound of Section 5.2 to provide a simple unprovability result for the variant of Cruanes' system considered in this chapter.

5.1 Cruanes' calculus

In this section we describe the variant of the system introduced by Cruanes in [Cru17] that we consider in this chapter. We will work with a slightly more abstract presentation of this system in order to make our results more robust against minor variations. A particularity of Cruanes' calculus is the explicit integration of clause splitting in the induction mechanism.

Remark 5.1.1. *The inductive theorem proving system described in [Cru17] is based on the extension of superposition by induction described in [Cru15]. However, [Cru15] uses a different induction mechanism based on smallest counterexamples.*

In Section 5.1.1 we briefly outline the idea underlying clause splitting and describe how we handle clause splitting. In Section 5.1.2 we recall the rules given in [Cru17] to handle the constructor symbols $0/0$ and $s/1$. Section 5.1.3 we describe the induction rule used by the system described in [Cru17]. Finally, in Section 5.1.4 we recall the cut-like rule that is described in [Cru17] and the variant that we use in the system considered in this chapter.

In this chapter we often work with induction formulas on several variables and with variable step width. For the sake of readability we introduce the notion of induction context defined below which binds together the information required to construct such induction formulas.

Definition 5.1.2. *Let $\mathcal{L}$ be a language, then an $\mathcal{L}$ induction context $\mathcal{I}$ is a triple of the form $(G, \vec{x}, \vec{m})$, where $G(\vec{x}, \vec{y})$ is an $\mathcal{L}$ formula, $\vec{x}$ a vector of variables, and $\vec{m} \in \mathbb{N}^{|\vec{x}|}$.*

5.1.1 Clause Splitting

Clause splitting is a technique used in saturation theorem proving to organize the search that relies on the observation that a theory $T + \varphi \vee \psi$ where φ and ψ are sentences is inconsistent if and only if $T + \varphi$ is inconsistent and $T + \psi$ is inconsistent. Thus in order to refute a set of clauses $\mathcal{C} \cup \{C \vee D\}$ where C and D are variable disjoint clauses, it suffices to refute the clause sets $\mathcal{C} \cup \{C\}$ and $\mathcal{C} \cup \{D\}$. Clause splitting allows to reduce the size of the clauses that a prover has to consider at the cost of introducing additional branching into the search space. There are two major ways to implement clause splitting: explicit and implicit clause splitting. Methods using explicit clause splitting keep a store of clause sets that need to be refuted. The implicit variant of clause splitting operates with a single clause set and splits the clause $C \vee D$ by introducing fresh predicate symbols $A/0$ and $B/0$ and replaces the clause $C \vee D$ with the clauses

$$C \leftarrow A, \quad D \leftarrow B, \quad A \vee B,$$

where $C \leftarrow A$ denotes the clause $C \vee \neg A$, see Definition 2.4.2. The theorem proving systems can now make use of the new predicates A and B to organize the search for a refutation. This type of splitting is most prominently used by the AVATAR architecture [Vor14] which uses a SAT solver to steer the search for a refutation. In practice the AVATAR method performs very well [Vor14]. We refer the reader to [Wei01] for a more detailed discussion of the variants of clause splitting.

Cruanes' calculus uses essentially the AVATAR clause splitting architecture and extends the splitting mechanism by integrating splitting directly into the induction rule. In this chapter we will also deal with a splitting mechanism that introduces new nullary predicate symbols but we handle the splitting very abstractly in the sense that we do not deal with how the splitting symbols are actually used by the underlying saturation system (that is we treat the splitting mechanism mostly as a black-box except for the introduction of split symbols.)

Let us start by introducing the split symbols that we will use throughout the chapter. There are mainly three sources of split symbols in the system that we will consider: the clause splitting rule, the lemma rule and the induction rule. We treat the symbols introduced by each of these rules separately because they are handled slightly differently. This is in particular the case for the split symbols that are introduced by the induction rule.

Definition 5.1.3 (Split symbols)**.** *Let φ be a sentence, then D_φ is a nullary predicate symbol, called a definition split symbol and L_φ is a nullary predicate symbol called a lemma split symbol. Let x be a variable, u a term, and $\mathcal{I}$ an induction context, then $\mathsf{I}^{x,u}_{\mathcal{I}}$ is a nullary predicate symbol, called an induction split symbol.*

We also introduce some notation to manipulate the split symbols of a given language.

Definition 5.1.4. *Let $\mathcal{L}$ be a language, then by $\mathcal{L}_{\mathsf{D}}$, $\mathcal{L}_{\mathsf{L}}$, and $\mathcal{L}_{\mathsf{I}}$ we denote the subsets of $\mathcal{L}$ sets consisting of the definition split symbols, the lemma split symbols, and the induction split symbols of $\mathcal{L}$, respectively. By $\mathcal{L}_{\mathsf{S}}$ we denote the set $\mathcal{L}_{\mathsf{D}} \cup \mathcal{L}_{\mathsf{L}} \cup \mathcal{L}_{\mathsf{I}}$. Let $\mathfrak{s} \in \{\mathsf{D}, \mathsf{L}, \mathsf{I}, \mathsf{S}\}$, then $\mathcal{L}_{\overline{\mathfrak{s}}} := \mathcal{L} \setminus \mathcal{L}_{\mathfrak{s}}$. We say that a language $\mathcal{L}$ is free of split symbols if $\mathcal{L}_{\mathsf{S}} = \varnothing$.*

We are now ready to define the rule that abstracts clause splitting mechanisms based on the introduction of new nullary predicate symbols.

Definition 5.1.5. *The clause splitting rule* $\mathrm{CSP}_{\mathrm{S}}$ *is given by*

$$\frac{C}{C' \vee \mathsf{D}_{\forall \vec{y} D} \quad D \leftarrow \mathsf{D}_{\forall \vec{y} D}}\ \mathrm{CSP}_{\mathrm{S}},$$

where $C(\vec{z})$, $C'(\vec{x})$, and $D(\vec{y})$ are clauses such that $\mathcal{L}(C'), \mathcal{L}(D) \subseteq \mathcal{L}(C)$, D is free of induction split symbols, and $(\forall \vec{z}) C$ is logically equivalent to $(\forall \vec{x}) C'(\vec{x}) \vee (\forall \vec{y}) D(\vec{y})$.

The clause splitting rule given in the definition above is an abstraction of clause splitting rules used in practical systems such as [Vor14] in the sense that the rule $\mathrm{CSP}_{\mathrm{S}}$ simulates the practical rules. Practical clause splitting rules usually differ from the rule $\mathrm{CSP}_{\mathrm{S}}$ in that they impose much stronger side conditions. For example, the separation of variable disjoint parts of a clause is usually done modulo commutativity of disjunction, renaming of variables, and symmetry of equality instead of full first-order equivalence. Moreover, practical clause splitting rules usually split only on maximal variable disjoint subclauses and variable-free literals are typically not abstracted by split symbols. For our purposes the more uniform and general formulation of the rule $\mathrm{CSP}_{\mathrm{S}}$ is more convenient and emphasizes the special behavior of induction split symbols.

We assume that the underlying saturation system makes use of the split symbols introduced, in particular, by the rule $\mathrm{CSP}_{\mathrm{S}}$ and therefore we will not explicitly introduce special inference rules for managing these symbols. In practice the splitting mechanism is of course coupled much more tightly with the rest of the system.

5.1.2 Injectivity, disjointness, and acyclicity

The method described in [Cru17] provides rules for dealing with the injectivity, disjointness, and acyclicity of inductive constructions. The idea underlying these rules is that they are more efficient to manage than the corresponding axioms. Similar rules as the ones given below are considered in [KRV17] and [Rob18]. Compared to the induction rule given in Section 5.1.3, the rules given in this section are not particularly interesting for our considerations. Nevertheless, we include these rules for the sake of completeness of the presentation.

Definition 5.1.6. *The disjointness rules* $\mathrm{DIS}^{-}_{\mathrm{S}}$, $\mathrm{DIS}^{+}_{\mathrm{S}}$ *and the injectivity rule* $\mathrm{INJ}_{\mathrm{S}}$ *are*

given by

$$\frac{0 = s(t) \vee C}{C}\ \mathrm{DIS}_{\mathrm{S}}^{+}, \quad \frac{0 \neq s(t) \vee C}{C}\ \mathrm{DIS}_{\mathrm{S}}^{-},$$
$$\frac{s(t) = s(t') \vee C}{t = t' \vee C}\ \mathrm{INJ}_{\mathrm{S}},$$

where t and t' are terms and C is a clause.

The acyclicity rules assert that an object (in our case a natural number) does not appear in its own construction. These rules are more manageable than the infinitely many instances of the corresponding axiom schema and thus help to improve the practical performance of an induction prover.

Definition 5.1.7. *The acyclicity rules $\mathrm{ACY}_{\mathrm{S}}^{+}$ and $\mathrm{ACY}_{\mathrm{S}}^{-}$ are given by*

$$\frac{t = s^{k+1}(u) \vee C}{C\sigma}\ \mathrm{ACY}_{\mathrm{S}}^{+}, \quad \frac{t \neq s^{k+1}(t) \vee C}{C}\ \mathrm{ACY}_{\mathrm{S}}^{-},$$

where t, u are terms, $k \in \mathbb{N}$, and σ is a unifier of t and u with $\mathcal{L}(t\sigma) = \mathcal{L}(t) \cup \mathcal{L}(u)$.

Remark 5.1.8. *In the formulation of the positive acyclicity rule in [Cru17] the constraint on the language of $t\sigma$ is not present. We have included this constraint so that the unifier does not extend the language.*

5.1.3 The induction rule

In this section we will define the induction rule as described in [Cru17]. As already mentioned before, the rule is similar to the induction rules discussed in Chapter 4. However, it is slightly more difficult to state because it makes use of a slightly more complicated induction schema and uses a custom splitting definitions. Hence, we start by stating the underlying induction schema and some related notions. Cruanes' method essentially uses an induction mechanism that is a variant of the diagonal big-step heterogeneous induction schema stated in Definition 4.2.23 for $\forall_1$ formulas.

The following definition introduces a notation for cover sets of natural numbers.

Definition 5.1.9. *Let $m \in \mathbb{N}$ and x a variable, then the set $\kappa_m(x)$ is given by*

$$\{0, \overline{1}, \ldots, \overline{m}, s^{m+1}(x)\}.$$

Let $\vec{m} = (m_1, \ldots, m_n) \in \mathbb{N}^n$ and $\vec{x} = (x_1, \ldots, x_n)$ a finite sequence of variables, then $\kappa_{\vec{m}}(\vec{x}) = \bigtimes_{i=1}^{n} \kappa_{m_i}(x_i)$.

We can now define the induction schema that underlies Cruanes' method. Since Cruanes' induction rule is rather complex, we introduce the required definitions and notations gradually. For the sake of legibility we introduce a separate definition for the premises of the induction axioms.

Definition 5.1.10. *Let $\mathcal{I} = (\varphi(\vec{x}, \vec{y}), \vec{x}, \vec{m})$ be an induction context with $\vec{x} = (x_1, \ldots, x_n)$ and $\vec{u} \in \kappa_{\vec{m}}(\vec{x})$, then the induction premise $IP^{\vec{u}}_{\mathcal{I}}(\vec{x}, \vec{y})$ is given by*

$$IP^{\vec{u}}_{\mathcal{I}}(\vec{x}, \vec{y}) = \begin{cases} (\forall \vec{y'})\varphi(\vec{x}, \vec{y'}) \to \varphi(s^{\vec{m}+1}(\vec{x}), \vec{y}) & \text{if } \vec{u} = s^{\vec{m}+1}(\vec{x}), \\ \varphi(\vec{u}, \vec{y}) & \text{otherwise.} \end{cases}$$

Furthermore, the induction premise $IP_{\mathcal{I}}(\vec{x}, \vec{y})$ is given by $\bigwedge_{\vec{u} \in \kappa_{\vec{m}}(\vec{x})} IP^{\vec{u}}_{\mathcal{I}}$ and the sentence $I_{\mathcal{I}}$ is given by $(\forall \vec{x})(\forall \vec{y}) IP_{\mathcal{I}} \to (\forall \vec{x})(\forall \vec{y})\varphi$

We will now give an example in order to illustrate the above definition.

Example 5.1.11. *Let $\mathcal{I} = (\varphi(x_1, x_2, \vec{y}), (x_1, x_2), (0, 1))$, then the formula $I_{\mathcal{I}}$ is given by*

$$(\forall \vec{x})(\forall \vec{y}) \left(\begin{array}{l} (\varphi(0, 0, \vec{y}) \wedge (\varphi(0, \overline{1}, \vec{y}) \wedge \varphi(0, s^2(x_2), \vec{y}) \\ \wedge \varphi(s(x_1), 0, \vec{y}) \wedge \varphi(s(x_1), \overline{1}, \vec{y}) \\ \wedge ((\forall \vec{y})\varphi(x_1, x_2, \vec{y}) \to \varphi(s(x_1), s^2(x_2), \vec{y}))) \end{array} \right) \to (\forall \vec{x})(\forall \vec{y})\varphi.$$

We will now gradually introduce some more notation that we will use to formulate Cruanes' induction rule. Cruanes' induction rule applies the induction schema given in Definition 5.1.10 only to quantifier-free formulas. In this case it is straightforward to Skolemize the induction axioms by Skolemizing the induction premises. We introduce some notation to refer to the Skolem constants corresponding to an induction axiom.

Definition 5.1.12. *Let $\mathcal{I} = (\varphi(\vec{x}, \vec{y}), \vec{x}, \vec{m})$ be an induction context with φ quantifier-free, then we denote by $\mu_{\mathcal{I}}$ and $\nu_{\mathcal{I}}$ the finite sequences of nullary Skolem constants such that*

$$sk^{\forall}((\forall \vec{x})(\forall \vec{y}) IP_{\mathcal{I}}) = IP_{\mathcal{I}}[\vec{x}/\mu_{\mathcal{I}}, \vec{y}/\nu_{\mathcal{I}}].$$

Cruanes' induction rule, just as the rules considered in Chapter 4, uses already derived clauses to construct an induction axiom. After that, the rule resolves the conclusion of the induction axiom with the clauses that were used to construct the induction formula. By resolving the Skolemized induction axiom $sk^{\exists}(I_{\mathcal{I}})$ with $\mathcal{I} = (\varphi(\vec{x}, \vec{y}), \vec{x}, \vec{m})$ against $\neg\varphi(\vec{t}, \vec{r})$ where $\vec{t}$ and $\vec{r}$ are finite sequences of terms, and by moving negations inwards over the conjunction, we obtain the following disjunction

$$\bigvee_{\vec{u} \in \kappa_{\vec{m}}(\vec{x})} \neg IP^{\vec{u}}_{\mathcal{I}}(\mu_{\mathcal{I}}, \nu_{\mathcal{I}}). \tag{5.1}$$

Now Cruanes' induction rule splits the above disjunction before clausification by introducing a split definition for each disjunct $\neg IP^{\vec{u}}_{\mathcal{I}}(\mu_{\mathcal{I}}, \nu_{\mathcal{I}})$ with $\vec{u} \in \kappa_{\vec{m}}(\vec{x})$.

Definition 5.1.13. *Let $\mathcal{I} = (\varphi, \vec{x}, \vec{m})$ be an induction context, then the induction premise definition $IPD_{\mathcal{I}}$ is given by*

$$\bigwedge_{\vec{u} \in \kappa_{\vec{m}}(\vec{x})} \left(\bigwedge_{i=1}^{|\vec{x}|} \mathsf{I}^{x_i, u_i}_{\mathcal{I}} \to \neg IP^{\vec{u}}_{\mathcal{I}}(\mu_{\mathcal{I}}, \nu_{\mathcal{I}}) \right).$$

Moreover, the induction premise split sentence $IPS_{\mathcal{I}}$ is given by

$$\bigwedge_{i=1}^{|\vec{x}|} \bigoplus_{u \in \kappa_{m_i}(x_i)} \mathsf{I}_{\mathcal{I}}^{x_i,u}.$$

Each conjunct of $IPD_{\mathcal{I}}$ corresponds to a definition for a disjunct $IP_{\mathcal{I}}^{\vec{u}}$ with $\vec{u} \in \kappa_{\vec{m}}(\vec{x})$ and the sentence $IPS_{\mathcal{I}}$ corresponds to the disjunction (5.1).

Example 5.1.14. *Let $C(x,y,z) \coloneqq x + (y + z) = (x + y) + z$ and $\mathcal{I} = (C, z, 0)$ be an induction context, then $IPS_{\mathcal{I}} = \mathsf{I}_{\mathcal{I}}^{z,0} \oplus \mathsf{I}_{\mathcal{I}}^{z,s(z)}$ and the sentence $IPD_{\mathcal{I}}$ consists of the conjuncts*

$$\mathsf{I}_{\mathcal{I}}^{z,0} \to (\nu_{\mathcal{I}})_0 + ((\nu_{\mathcal{I}})_1 + 0) \not\approx ((\nu_{\mathcal{I}})_0 + (\nu_{\mathcal{I}})_1) + 0,$$
$$\mathsf{I}_{\mathcal{I}}^{z,s(z)} \to \neg \begin{pmatrix} (\forall x)(\forall y)(x + (y + (\mu_{\mathcal{I}})_0) = (x + y) + (\mu_{\mathcal{I}})_0) \to \\ (\nu_{\mathcal{I}})_0 + ((\nu_{\mathcal{I}})_1 + s((\mu_{\mathcal{I}})_0)) = ((\nu_{\mathcal{I}})_0 + (\nu_{\mathcal{I}})_1) + s((\mu_{\mathcal{I}})_0) \end{pmatrix}.$$

We can now define the variant of the induction schema given in Definition 5.1.10 that includes Skolemization and splitting.

Definition 5.1.15. *Let $\mathcal{L}$ be a first-order language, then the theory $\mathcal{L}$-*CSIND *is axiomatized by the sentences $I_{\mathcal{I}}^{CS}$ given by*

$$IPS_{\mathcal{I}} \wedge (\neg IPD_{\mathcal{I}} \to \forall \vec{x}\vec{y}\varphi),$$

where $\mathcal{I} = (\varphi, \vec{x}, \vec{m})$ is an induction context and $\varphi(\vec{x}, \vec{y})$ is a quantifier-free $\mathcal{L}$ formula.

With this induction schema in mind it is easy to state the induction rule of the system described in [Cru17].

Definition 5.1.16. *The induction rule* CIND_{S} *is given by*

$$\frac{C_1(\vec{c}, \vec{d}, \vec{z}) \vee A_1 \quad \dots \quad C_n(\vec{c}, \vec{d}, \vec{z}) \vee A_n}{cnf\,(IPS_{\mathcal{I}}) \cup \left(cnf\,(IPD_{\mathcal{I}}) \vee \bigvee_{i=1,\dots,n} A_i \right)}\ \text{CIND}_{\text{S}},$$

where $\mathcal{I} = (\neg \bigwedge_{i=1}^{n} C_i(\vec{x}, \vec{y}, \vec{z}), \vec{x}, \vec{m})$ is an induction context, $C_1, \dots, C_n$ are clauses free of induction split symbols, and $\vec{c}$, $\vec{d}$ are finite sequences of nullary function symbols with $|\vec{c}| = |\vec{x}|$ and $|\vec{d}| = |\vec{y}|$.

Similarly to the rules discussed in Chapter 4 the induction rule used in Cruanes' induction rule is "analytic" in the sense that it only carries out induction using clauses that have been previously derived.

Remark 5.1.17. *A particularity of the induction rule given in Definition 5.1.16 is that in order to refute the clauses $C_1(\vec{c}, \vec{d}, \vec{z})$, …, $C_n(\vec{c}, \vec{d}, \vec{z})$ the calculus will not try to infer $(\forall \vec{x})(\forall \vec{y})(\exists \vec{z})(\neg \bigwedge_{i=1}^{n} C_i)$, but instead it tries to infer the stronger $(\forall \vec{x})(\forall \vec{y})(\forall \vec{z})(\neg \bigwedge_{i=1}^{n} C_i)$.*

The following example demonstrates an application of the induction rule $\mathrm{CIND}_{\mathrm{S}}$.

Example 5.1.18. *Let* $C(x,y,z) := x + (y + z) = (x + y) + z$ *and* $\mathcal{C}_0 = cnf(B) \cup \{\neg C(c_0,c_1,c_2)\}$, *where* $sk^{\forall}((\forall x)(\forall y)(\forall z)C) = C(c_0,c_1,c_2)$. *Then by an application of the rule* $\mathrm{CIND}_{\mathrm{S}}$ *with* $\mathcal{I} = (C,z,0)$ *to the clause* $\neg C(c_0,c_1,c_2) \in \mathcal{C}_0$ *we obtain the clause set* $\mathcal{C}_1 = \mathcal{C}_0 \cup cnf(IPS_{\mathcal{I}}) \cup cnf(IPD_{\mathcal{I}})$. *Hence,* $\mathcal{C}_1 \models \mathsf{I}^{z,0}_{\mathcal{I}} \oplus \mathsf{I}^{z,s(z)}_{\mathcal{I}}$. *If* $\mathcal{C}_1 \models \mathsf{I}^{z,0}_{\mathcal{I}}$, *then* $\mathcal{C}_1 \models (\nu_{\mathcal{I}})_0 + ((\nu_{\mathcal{I}})_1 + 0) \not\approx ((\nu_{\mathcal{I}})_0 + (\nu_{\mathcal{I}})_1) + 0$. *Since* $\mathcal{C}_1 \models x + 0 = x$, *we obtain* $\mathcal{C}_1 \models (\nu_{\mathcal{I}})_0 + (\nu_{\mathcal{I}})_1 \not\approx (\nu_{\mathcal{I}})_0 + (\nu_{\mathcal{I}})_1$, *that is,* $\mathcal{C}_1 \models \bot$. *Hence,* $\mathcal{C}_1 \models \mathsf{I}^{z,s(z)}_{\mathcal{I}}$ *and therefore*

$$\mathcal{C}_1 \models x + (y + (\mu_{\mathcal{I}})_0) = (x + y) + (\mu_{\mathcal{I}})_0, \tag{$*$}$$

$$\mathcal{C}_1 \models (\nu_{\mathcal{I}})_0 + ((\nu_{\mathcal{I}})_1 + s((\mu_{\mathcal{I}})_0)) \not\approx ((\nu_{\mathcal{I}})_0 + (\nu_{\mathcal{I}})_1) + s((\mu_{\mathcal{I}})_0). \tag{$\star$}$$

Thus, by $(\star)$ *we have* $\mathcal{C}_1 \models s((\nu_{\mathcal{I}})_0 + ((\nu_{\mathcal{I}})_1 + (\mu_{\mathcal{I}})_0)) \not\approx s((\nu_{\mathcal{I}})_0 + (\nu_{\mathcal{I}})_1 + (\mu_{\mathcal{I}})_0)$. *Hence, by* $(*)$ *we obtain* $\mathcal{C}_1 \models s((\nu_{\mathcal{I}})_0 + ((\nu_{\mathcal{I}})_1 + (\mu_{\mathcal{I}})_0)) \not\approx s((\nu_{\mathcal{I}})_0 + ((\nu_{\mathcal{I}})_1 + (\mu_{\mathcal{I}})_0))$. *Thus* $\mathcal{C}_1 \models \bot$.

5.1.4 The lemma rule

In order to compensate for the analyticity inherent to the induction rule $\mathrm{CIND}_{\mathrm{S}}$ the calculus given in [Cru17] provides a cut-like rule that allows to introduce new syntactic material into the search space. In practice this rule is used as a uniform interface for the integration of heuristics that guess induction formulas, see [Cru17]. We start by considering a general formulation of the lemma rule as given in [Cru15; Cru17].

Definition 5.1.19. *The lemma rule* LEM *is given by*

$$\frac{C_1 \quad \dots \quad C_n}{\big(cnf(sk^{\exists}(\varphi)) \leftarrow \mathsf{L}_{\varphi}\big) \cup \big(cnf(sk^{\exists}(\neg\varphi)) \leftarrow \neg\mathsf{L}_{\varphi}\big)}\ \mathrm{LEM}$$

where φ *is an* $L(C_1,\dots,C_n)$ *sentence free of induction split symbols.*

In this thesis we concentrate on a family of methods described in [Cru17, Section 6] that apply LEM to $\forall_1$ formulas only.

Definition 5.1.20. *The lemma rule for* $\forall_1$ *sentences* $\mathrm{LEM}^{\forall_1}_{\mathrm{S}}$ *is given by*

$$\frac{C_1 \quad \dots \quad C_n}{(cnf(\varphi) \leftarrow \mathsf{L}_{\varphi}) \cup \big(cnf(sk^{\exists}(\neg\varphi)) \leftarrow \neg\mathsf{L}_{\varphi}\big)}\ \mathrm{LEM}^{\forall_1}_{\mathrm{S}},$$

where φ *is a* $\forall_1(\mathcal{L}(C_1,\dots,C_n))$ *sentence free of induction split symbols.*

We have now introduced all the inference and simplification rules that we need to formulate Cruanes' system as described in [Cru17]. In the following section we will provide an upper bound on the logical strength of a sound saturation system extended by the rules introduced in this section.

5.2 An upper bound

In this section we will provide a theory that serves as an upper bound on the logical strength of a sound saturation system extended by the rules introduced in the previous section. As in Chapters 3 and 4, the upper bound is Skolem-free in the $\exists_2$ setting and is thus particularly useful for the development of simple unprovability results. In Section 5.3 we will show that the upper bound derived in this section even gives rise to a a completeness result in the $\exists_2$ setting. Finally, in Section 5.4 we make use of the upper bound to provide a simple unprovability result.

The derivation of the upper bound proceeds essentially in the same way as the upper bounds derived in Chapters 3 and 4, except that we need some additional steps to deal with the split symbols. Let us start by introducing some notation for the extension of saturation systems by the rules introduced above.

Definition 5.2.1. *Let $\mathcal{S}$ be a saturation system then by $C_{\mathcal{S}}$ we denote the saturation system $\mathcal{S}$ extended by the rules* $\mathrm{CSP_S}$, $\mathrm{CIND_S}$, $\mathrm{LEM_S^{\forall_1}}$, $\mathrm{DIS_S^-}$, $\mathrm{DIS_S^+}$, $\mathrm{INJ_S}$, $\mathrm{ACY_S^-}$, *and* $\mathrm{ACY_S^+}$.

We obtain the upper bound in several steps. Dealing directly with the deductions of the system $C_{\mathcal{S}}$, where $\mathcal{S}$ is a sound saturation system, is tedious because of the many rules that are involved. Hence, we start by simulating the system $C_{\mathcal{S}}$ in an iterative construction similar to the ones used in [HV23]. This also provides us with a more convenient formulation of the languages that are generated by the deductions of the system $C_{\mathcal{S}}$. In the next step we eliminate the induction split symbols and simplify the induction schema. After that we unfold the definitions introduced by the rules $\mathrm{CSP_S}$ and $\mathrm{LEM_S^{\forall_1}}$.

Definition 5.2.2. *Let Φ be a set of formulas, then the theory* Φ-LDEF *is axiomatized by the sentences $\mathsf{L}_{(\forall\vec{x})\varphi} \leftrightarrow (\forall\vec{x})\varphi$ for $\varphi(\vec{x}) \in \Phi$. Similarly, the theory* Φ-DDEF *is axiomatized by the sentences $\mathsf{D}_{(\forall\vec{x})\varphi} \leftrightarrow (\forall\vec{x})\varphi$ for $\varphi(\vec{x}) \in \Phi$.*

The operator given below is similar to the ones used in [HV23]. It allows us primarily to abstract the saturation system $C_{\mathcal{S}}$ and to express the language generated by deductions of this system.

Definition 5.2.3. *Let T be a theory, then we define*

$$\begin{aligned} T_C(T) \coloneqq T &+ \mathrm{Clause}(\mathcal{L}(T)_{\bar{\mathsf{l}}})\text{-DDEF} \\ &+ sk^{\exists}(\forall_1(\mathcal{L}(T)_{\bar{\mathsf{l}}})\text{-LDEF}) \\ &+ \mathcal{L}(T)_{\bar{\mathsf{l}}}\text{-CSIND}. \end{aligned}$$

By $T_C^i(T)$ we denote the i-th iteration of the operator T_C on T. Furthermore, we define

$$T_C^{\omega}(T) \coloneqq \bigcup_{i<\omega} T_C^i(T).$$

It is straightforward but tedious to simulate deductions in Cruanes system with the operator introduced above.

Lemma 5.2.4. *Let $\mathcal{S}$ be a sound saturation system and $\mathcal{C}_0, \ldots, \mathcal{C}_n$ be a $C_\mathcal{S}$ deduction, then $\mathcal{A}_0 + \mathrm{ACY} + T^\omega_C(\mathcal{C}_0) \vdash \mathcal{C}_n$ and $\mathcal{L}(\mathcal{C}_n) \subseteq \mathcal{L}(T^\omega_C(\mathcal{C}_0))$.*

Proof. We proceed by induction on n. The base case $n = 0$ is trivial, since $T^\omega_C(\mathcal{C}_0) \vdash \mathcal{C}_0$. For the induction step we assume that $\mathcal{L}(\mathcal{C}_n) \subseteq \mathcal{L}(T^\omega_C(\mathcal{C}_0))$ and

$$\mathcal{A}_0 + \mathrm{ACY} + T^\omega_C(\mathcal{C}_0) \vdash \mathcal{C}_n.$$

Now we distinguish several cases depending on how the clause set $\mathcal{C}_{n+1}$ was obtained.

If $\mathcal{C}_{n+1}$ is derived by an application of $\mathrm{LEM}^{\forall_1}_\mathrm{S}$, then

$$\mathcal{C}_{n+1} = \mathcal{C}_n \cup \left(cnf(sk^\exists(\varphi)) \leftarrow \mathsf{L}_\varphi \right) \cup \left(cnf(\neg sk^\exists(\varphi)) \leftarrow \neg \mathsf{L}_\varphi \right),$$

where φ is a $\forall_1(\mathcal{L}(\mathcal{C}_n)_{\overline{\mathsf{l}}})$ formula. By the induction hypothesis φ is a $\mathcal{L}(T^\omega_C(\mathcal{C}_0))_{\overline{\mathsf{l}}}$ formula, thus, $T^\omega_C(\mathcal{C}_0) \vdash sk^\exists(\mathsf{L}_\varphi \leftrightarrow \varphi)$. Therefore, $\mathcal{L}(\mathcal{C}_{n+1}) \subseteq \mathcal{L}(T^\omega_C(T))$ and $\mathcal{A}_0 + \mathrm{ACY} + T^\omega_C(T) \vdash \mathcal{C}_{n+1}$.

If $\mathcal{C}_{n+1}$ is obtained by an application of the rule CIND_S, then

$$\mathcal{C}_{n+1} = \mathcal{C}_n \cup cnf\left(IPS_\mathcal{I}\right) \cup \left(cnf(IPD_\mathcal{I}) \vee \bigvee_{i=1}^{m} A_i \right)$$

where $\mathcal{I} = (\neg\varphi, \vec{x}, \vec{m})$ is an induction context with $\varphi = \bigwedge_{i=1}^m C_i(\vec{x}, \vec{y}, \vec{z})$, C_i is a clause free of induction split symbols, $A_i(\vec{z})$ is a clause, $C_i(\vec{c}, \vec{d}, \vec{z}) \vee A_i \in \mathcal{C}_n$ for $i = 1, \ldots, m$ and $\vec{c}, \vec{d}$ are finite sequences of nullary function symbols. By the induction hypothesis we have in particular, $\mathcal{L}(C_i(\vec{c}, \vec{d}, \vec{z}) \vee A_i) \subseteq \mathcal{L}(T^\omega_C(\mathcal{C}_0))$. Hence, $I^{CS}_\mathcal{I} \in T^\omega_C(\mathcal{C}_0)$ and therefore $\mathcal{L}(\mathcal{C}_{n+1}) \subseteq \mathcal{L}(T^\omega_C(T))$. Moreover, we have $\mathcal{A}_0 + \mathrm{ACY} + T^\omega_C(\mathcal{C}_0) \vdash C_i(\vec{c}, \vec{d}, \vec{z}) \vee A_i$ for $i = 1, \ldots, m$. Now work in $\mathcal{A}_0 + \mathrm{ACY} + T^\omega_C(\mathcal{C}_0)$. By the above we already have $IPS_\mathcal{I}$ and it remains to show $IPD_\mathcal{I} \vee \bigvee_{i=1}^m A_i$. Let $\vec{z}$ be arbitrary but fixed. If A_i for some $i \in \{1, \ldots, m\}$, then we are done. Otherwise, assume not A_i for $i = 1, \ldots, m$. We have $\bigwedge_{i=1}^m C_i(\vec{c}, \vec{d}, \vec{z})$. Hence, not $(\forall\vec{x})(\forall\vec{y})(\forall\vec{z})\neg\bigwedge_{i=1}^m C_i$. Therefore, we have $IPD_\mathcal{I}$. Thus $\mathcal{A}_0 + \mathrm{ACY} + T^\omega_C(\mathcal{C}_0) \models \mathcal{C}_{n+1}$.

If $\mathcal{C}_{n+1}$ is obtained by an application of the rule CSP_S, then

$$\mathcal{C}_{n+1} = \mathcal{C}_n \cup \{C' \vee \mathsf{D}_{(\forall\vec{y})D}, D \leftarrow \mathsf{D}_{(\forall\vec{y})D}\},$$

where $C(\vec{z}) \in \mathcal{C}_n$ and $\vdash (\forall\vec{z})C \leftrightarrow (\forall\vec{x})C' \vee (\forall\vec{y})D$ with $C'(\vec{x}), D(\vec{y})$ clauses such that $\mathcal{L}(C'), \mathcal{L}(D) \subseteq \mathcal{L}(C)$. Since in particular $\mathcal{L}(D) \subseteq \mathcal{L}(T^\omega_C(\mathcal{C}_0))_{\overline{\mathsf{l}}}$, we have $\mathsf{D}_{(\forall\vec{y})D} \leftrightarrow (\forall\vec{y})D \in T^\omega_C(\mathcal{C}_0)$. Thus $\mathcal{L}(\mathcal{C}_{n+1}) \subseteq \mathcal{L}(T^\omega_C(\mathcal{C}_0))$. By the induction hypothesis, we have, in particular, $\mathcal{A}_0 + \mathrm{ACY} + T^\omega_C(\mathcal{C}_0) \vdash C$. Hence, $\mathcal{A}_0 + \mathrm{ACY} + T^\omega_C(\mathcal{C}_0) \vdash C' \vee (\forall\vec{y})D$. Therefore, since C' and D are variable disjoint we have $\mathcal{A}_0 + \mathrm{ACY} + T^\omega_C(\mathcal{C}_0) \vdash C' \vee \mathsf{D}_{(\forall\vec{y})D}$.

If $\mathcal{C}_{n+1}$ is obtained by an application of INJ_S, then there is a clause $s(t) = s(t') \vee C \in \mathcal{C}_n$ such that $\mathcal{C}_{n+1} = (\mathcal{C}_n \setminus \{s(t) = s(t') \vee C\}) \cup \{t = t' \vee C\}$. By the induction hypothesis it

is obvious that $\mathcal{L}(\mathcal{C}_{n+1}) \subseteq \mathcal{L}(T_C^\omega(\mathcal{C}_0))$. Moreover, we have $\mathcal{A}_0 + \text{ACY} + T_C^\omega(\mathcal{C}_0) \vdash s(t) = s(t') \vee C$. Since $\mathcal{A}_0 \vdash s(x) = s(y) \to x = y$, we obtain $\mathcal{A}_0 + \text{ACY} + T_C^\omega(\mathcal{C}_0) \vdash t = t' \vee C$.

If $\mathcal{C}_{n+1}$ is obtained by an application of DIS_S^+, then there exists a clause $0 = s(t) \vee C \in \mathcal{C}_n$ and $\mathcal{C}_{n+1} = (\mathcal{C}_n \setminus \{0 = s(t) \vee C\}) \cup \{C\}$. By the induction hypothesis we readily have $\mathcal{L}(\mathcal{C}_{n+1}) \subseteq \mathcal{L}(T_C^\omega(\mathcal{C}_0))$. Moreover $\mathcal{A}_0 + \text{ACY} + T_C^\omega(\mathcal{C}_0) \vdash 0 = s(t) \vee C$. Since, $\mathcal{A}_0 \models 0 \neq s(x)$, we thus have $\mathcal{A}_0 + \text{ACY} + T_C^\omega(\mathcal{C}_0) \models C$.

If $\mathcal{C}_{n+1}$ is obtained by an application of ACY_S^+, then there exists a clause $t = s^{k+1}(u) \vee C \in \mathcal{C}_n$ with $k \in \mathbb{N}$ such that $\mathcal{C}_{n+1} = \big(\mathcal{C}_n \setminus \{t = s^{k+1}(u)\}\big) \cup C\sigma$, where σ is a unifier of t and u and $\mathcal{L}(\sigma(t)) \subseteq \mathcal{L}(t) \cup \mathcal{L}(u)$. Hence, $\mathcal{L}(\mathcal{C}_{n+1}) \subseteq \mathcal{L}(T_C^\omega(\mathcal{C}_0))$. By the induction hypothesis we have in particular $\mathcal{A}_0 + \text{ACY} + T_C^\omega(\mathcal{C}_0) \vdash t\sigma = s^{k+1}(u\sigma) \vee C\sigma$. Since $\mathcal{A}_0 + \text{ACY} + T_C^\omega(\mathcal{C}_0) \vdash x \neq s^{k+1}(x)$, we have $\mathcal{A}_0 + \text{ACY} + T_C^\omega(\mathcal{C}_0) \vdash C\sigma$.

If $\mathcal{C}_{n+1}$ is obtained by an application of DIS_S^-, ACY_S^-, or $\mathcal{S}$, then $\mathcal{C}_{n+1} \vdash \mathcal{C}_n$ and $\mathcal{L}(\mathcal{C}_{n+1}) \subseteq \mathcal{L}(\mathcal{C}_n)$. Hence, we are done by the induction hypothesis. □

The iterative construction of $T_C^\omega(\cdot)$ is mostly useful to build up the languages that can be generated by $C_\mathcal{S}$ where $\mathcal{S}$ is a sound saturation system.

Definition 5.2.5. *Let T be a theory, then we let $\mathcal{L}_C^T = \mathcal{L}(T_C^\omega(T))_\text{ī}$.*

Lemma 5.2.6. *Let T be a theory, then*

$$T + sk^\omega\left(\mathcal{L}_C^T\right)\text{-SA} + \text{Clause}(\mathcal{L}_C^T)\text{-DDEF} + \forall_1(\mathcal{L}_C^T)\text{-LDEF} + \mathcal{L}_C^T\text{-CSIND} \vdash T_C^\omega(T).$$

Proof. We proceed by induction on i and show that $T + sk^\omega\left(\mathcal{L}_C^T\right)\text{-SA} + \text{Clause}(\mathcal{L}_C^T)\text{-DDEF} + \forall_1(\mathcal{L}_C^T)\text{-LDEF} + \mathcal{L}_C^T\text{-CSIND} \vdash T_C^i(T)$. The base case is trivial, since $T_C^0(T) = T$. For the induction step we have $T_C^{i+1}(T) = T_C(T_C^i(T))$. By the induction hypothesis we already have

$$T + sk^\omega\left(\mathcal{L}_C^T\right)\text{-SA} + \text{Clause}(\mathcal{L}_C^T)\text{-DDEF} + \forall_1(\mathcal{L}_C^T)\text{-LDEF} + \mathcal{L}_C^T\text{-CSIND} \vdash T_C^i(T).$$

Since $\mathcal{L}(T_C^i(T))_\text{ī} \subseteq \mathcal{L}_C^T$, it is obvious that

$$\begin{gathered}\forall_1(\mathcal{L}_C^T)\text{-LDEF} + sk^\omega\left(\mathcal{L}_C^T\right)\text{-SA} \vdash sk^\exists(\forall_1(\mathcal{L}_C^T)\text{-LDEF}),\\ \text{Clause}(\mathcal{L}_C^T)\text{-DDEF} \vdash \text{Clause}(\mathcal{L}(T_C^i(T))_\text{ī})\text{-DDEF},\\ \mathcal{L}_C^T\text{-CSIND} \vdash \mathcal{L}(T_C^i(T))_\text{ī}\text{-CSIND}.\end{gathered}$$

Hence, the claim follows immediately from the induction hypothesis. □

The next step of the derivation of the upper bound for Cruanes' system is the simplification of the induction schema. In a first step we will eliminate the induction split symbols. After that we deskolemize the induction axioms and reformulate the induction axiom. Finally, we introduce induction parameters in order to eliminate Skolem symbols introduced during the saturation process.

Lemma 5.2.7 (Elimination of induction split symbols)**.** *Let T be a theory free of induction split symbols, $\mathcal{L} \supseteq \mathcal{L}_C^T$ a language, and M an $\mathcal{L}$ structure such that $M \models$*

$sk^{\exists}(\mathcal{L}_C^T$-CIND$)$, then there exists a structure M' of M such that $M|\mathcal{L}_{\mathsf{I}} = M'|\mathcal{L}_{\mathsf{I}}$ and $M' \models \mathcal{L}_C^T$-CSIND

Proof. Let $M|\mathcal{L}_{\mathsf{I}} = (D, I)$, then we will expand the interpretation I to I' as follows. Let $\mathcal{I} = (\psi(\vec{x}, \vec{y}), \vec{x}, \vec{m})$ be a $\mathcal{L}_C^T$ induction context where ψ is quantifier-free. We have to provide an interpretation for the symbols $\mathsf{I}_{\mathcal{I}}^{x_i,u}$ for $i = 1, \ldots, |\vec{x}|$ and $u \in \kappa_{m_i}(x_i)$.

We start with the case $M \models (\forall\vec{x})(\forall\vec{y})\psi$. For $i = 1, \ldots, |\vec{x}|$ and $u \in \kappa_{m_i}(x_i)$ we let

$$I'\left(\mathsf{I}_{\mathcal{I}}^{x_i,u}\right) = \begin{cases} 1 & \text{if } u = 0, \\ 0 & \text{otherwise} \end{cases}.$$

Since $M \models (\forall\vec{x})(\forall\vec{y})\psi$ and ψ is a $\mathcal{L}_C^T$ formula, we have $(D, I') \models \neg IPD_{\mathcal{I}} \to (\forall\vec{x})(\forall\vec{y})\psi$. Moreover, by the definition of I' we also have $(D, I') \models IPS_{\mathcal{I}}$. Hence we are done.

Otherwise, if $M \not\models (\forall\vec{x})(\forall\vec{y})\psi$ there exists $\vec{u} \in \kappa_{\vec{m}}(\vec{x})$ such that $M \not\models IP_{\mathcal{I}}^{\vec{u}}(\mu_{\mathcal{I}}, \nu_{\mathcal{I}})$. For $i = 1, \ldots, |\vec{x}|$ and $v_i \in \kappa_{m_i}(x_i)$ we now define

$$\left(\mathsf{I}_{\mathcal{I}}^{x_i,v_i}\right)^{M'} := \begin{cases} 1 & \text{if } v_i = u_i \\ 0 & \text{otherwise} \end{cases}.$$

Clearly, we then have $M' \models IPS_{\mathcal{I}}$. Hence it remains to show that $(D, I') \models IPD_{\mathcal{I}}$. Now let $\vec{v} \in \kappa_{\vec{m}}(\vec{x})$. We need to consider two cases. If $\vec{v} = \vec{u}$, then $(D, I') \not\models IP_{\mathcal{I}}^{\vec{v}}(\mu_{\mathcal{I}}, \nu_{\mathcal{I}})$. Therefore, $(D, I') \models \bigwedge_{i=1}^{|\vec{x}|} \mathsf{I}_{\mathcal{I}}^{x_i,v_i} \to \neg IP_{\mathcal{I}}^{\vec{v}}(\mu_{\mathcal{I}}, \nu_{\mathcal{I}})$. If $\vec{v} \neq \vec{u}$, then let $v_{i_0} \neq u_{i_0}$. Then we have $(D, I') \not\models \mathsf{I}_{\mathcal{I}}^{x_i,v_i}$. Thus $(D, I') \models \bigwedge_{i=1}^{|\vec{x}|} \mathsf{I}_{\mathcal{I}}^{x_i,v_i} \to \neg IP_{\mathcal{I}}^{\vec{v}}(\mu_{\mathcal{I}}, \nu_{\mathcal{I}})$. Hence, $(D, I') \models IPS_{\mathcal{I}} \wedge (\neg IPD_{\mathcal{I}} \to (\forall\vec{x})(\forall\vec{y})\psi)$. □

The following two lemmas allow us to reformulate Cruanes' induction schema essentially as parameter-free induction on $\forall_1$ formulas.

Lemma 5.2.8. *Let $\varphi(\vec{x}, \vec{z})$ be a formula and $\vec{m} \in \mathbb{N}$ with $|\vec{m}| = |\vec{x}|$, then* Pred $\vdash \forall\vec{x}\varphi(\vec{x}, \vec{z}) \leftrightarrow \forall\vec{x} \bigwedge_{\vec{u} \in \kappa_{\vec{m}}(\vec{x})} \varphi(\vec{u}, \vec{z})$.

Proof. A straightforward induction on $|\vec{x}|$. □

Lemma 5.2.9. *Let $\mathcal{L}$ be a language, then* Pred $+ \forall_1(\mathcal{L})$-IND$^- \vdash \mathcal{L}$-CIND.

Proof. In the light of Lemmas 4.2.3, 4.2.26, and 4.2.27 it suffices to show

$$\text{Pred} + \forall_1(\mathcal{L})\text{-BS}_{\text{het}}\text{Diag}_{\omega}\text{IND}^- \vdash \mathcal{L}\text{-CIND}.$$

Let $\mathcal{I} = (\varphi(\vec{x}, \vec{y}), \vec{x}, \vec{m})$ be an induction context where φ is a quantifier-free formula. Assume $IP_{\mathcal{I}}^{\vec{u}}$ for all $\vec{u} \in \kappa_{\vec{m}}(\vec{x})$. We proceed with $I_{\vec{x},\vec{m}}^{\text{Diag}}(\forall\vec{y})\varphi$. Let $i \in \{1, \ldots, |\vec{x}|\}$ and $u_i \in \kappa_{m_i}(x_i)$ with $u_i \neq s^{m_i}(x_i)$. By the assumption we have $\varphi(\vec{v}_{<i}, u_i, \vec{v}_{>i})$ for all $\vec{v}_{<i} \in \kappa_{\vec{m}_{<i}}(\vec{x}_{<i})$ and $\vec{v}_{>i} \in \kappa_{\vec{m}_{>i}}(\vec{x}_{>i})$. Hence, by Lemma 5.2.8 we obtain $\varphi(\vec{x}_{<i}, u_i, \vec{x}_{>i})$. Now assume $(\forall\vec{y})\varphi(\vec{x}, \vec{y})$ and let $\vec{y}$ be arbitrary, then by the assumption $IP_{\mathcal{I}}^{\vec{x}_i, s^{\vec{m}_i}(\vec{x}_i)}$ we obtain $\varphi(s^{\vec{m}_i}(\vec{x}_i), \vec{y})$. Hence, we obtain $(\forall\vec{x})(\forall\vec{y})\varphi(\vec{x}, \vec{y})$. □

Let us now show that we can decompose the language generated by Cruanes' system so as to separate in particular the Skolem symbols that may be introduced during saturation.

Lemma 5.2.10. *Let T be a theory, then*

$$\mathcal{L}(T_C^\omega(T)) = \left(\mathcal{L}_C^T\right)_{\mathsf{I}} \cup \left(\mathcal{L}_C^T\right)_{\mathsf{D}} \cup \left(\mathcal{L}_C^T\right)_{\mathsf{L}} \cup \mathcal{L}(T) \cup \mathcal{L}_0 \cup \Sigma_C^T,$$

where Σ_C^T is a set of nullary function symbols.

Proof. We proceed by induction on i and show that there exists a set of nullary function symbols Σ_i such that

$$\mathcal{L}(T_C^i(T)) \subseteq \mathcal{L}(T_C^i(T))_{\mathsf{I}} \cup \mathcal{L}(T_C^i(T))_{\mathsf{D}} \cup \mathcal{L}(T_C^i(T))_{\mathsf{L}} \cup \mathcal{L}(T) \cup \mathcal{L}_0 \cup \Sigma_i.$$

The base case $n = 0$ is obvious since $T_C^0(T) = T$. For the induction step consider the theory $T_C^{i+1}(T) = T_C(T_C^i(T))$. The axioms of Clause$(\mathcal{L}(T_C^i(T))_{\bar{\mathsf{I}}})$-DDEF consist of symbols of $\mathcal{L}(T_C^i(T))_{\bar{\mathsf{I}}}$ and definition split symbols. The axioms of $sk^\exists(\forall_1(\mathcal{L}(T_C^i(T))_{\bar{\mathsf{I}}})$-LDEF) consist of $\mathcal{L}(T_C^i(T))_{\bar{\mathsf{I}}}$ symbols, lemma split symbols, and nullary Skolem symbols arising from the $\forall$ Skolemization of $\forall_1$ formulas. Finally, the axioms of $\mathcal{L}(T_C^i(T))_{\bar{\mathsf{I}}}$-CSIND consist of $\mathcal{L}(T_C^i(T))_{\bar{\mathsf{I}}}$ symbols, induction split symbols, the symbols $0/0$ and $s/1$, and the nullary Skolem symbols in $\mu_{\mathcal{I}}$ and in $\nu_{\mathcal{I}}$ for induction contexts $\mathcal{I} = (\varphi(x, \vec{y}), \vec{x}, \vec{m})$ with φ a quantifier-free $\mathcal{L}(T_C^i(T))_{\bar{\mathsf{I}}}$. Thus, we are done by the induction hypothesis. □

We can now absorb the Skolem symbols that are generated during the saturation process in Cruanes' system by induction parameters. Moreover, the split symbols that may occur in the induction are also easily seen to be absorbed by the induction axioms.

Lemma 5.2.11. $\forall_1(\mathcal{L}(T) \cup \mathcal{L}_0)$-IND $\vdash \mathcal{L}_C^T$-CIND.

Proof. By Lemma 5.2.9 we have Pred $+ \forall_1(\mathcal{L}_C^T)$-IND$^- \vdash \mathcal{L}_C^T$-CIND. By Lemma 5.2.10 we have

$$\mathcal{L}_C^T = \mathcal{L}(T) \cup \mathcal{L}_0 \cup \left(\mathcal{L}_C^T\right)_{\mathsf{D}} \cup \left(\mathcal{L}_C^T\right)_{\mathsf{L}} \cup \Sigma_C^T,$$

where Σ_C^T is a set of nullary function Symbols. Hence, it is obvious that

$$\forall_1 (\mathcal{L}(T) \cup \mathcal{L}_0)\text{-IND} \vdash \mathcal{L}_C^T\text{-CIND}.$$ □

This completes the first step of the derivation of the upper bound. We summarize the results so far in the following proposition.

Proposition 5.2.12. *Let $\mathcal{S}$ be a sound saturation system, and T a theory free of induction split symbols. If $C_{\mathcal{S}}$ refutes the clause set cnf$(sk^\exists(T))$, then the theory*

$$\begin{aligned}\mathcal{A}_0 + sk^\exists(T) + sk^\omega\left(\mathcal{L}_C^{sk^\exists(T)}\right)\text{-SA} + \text{Clause}(\mathcal{L}_C^{sk^\exists(T)})\text{-DDEF} \\ + \forall_1\left(\mathcal{L}_C^{sk^\exists(T)}\right)\text{-LDEF} + \forall_1(\mathcal{L}(sk^\exists(T) \cup \mathcal{L}_0))\text{-IND}\end{aligned}$$

is inconsistent.

Proof. Assume that $\mathcal{S}$ refutes $cnf(sk^{\exists}(T))$, then by Lemma 5.2.4 the theory $\mathcal{A}_0 + \mathrm{ACY} + T_C^{\omega}(sk^{\exists}(T))$ is inconsistent. Hence, by Lemma 5.2.6 the following theory is inconsistent as well

$$\mathcal{A}_0 + \mathrm{ACY} + sk^{\exists}(T) + sk^{\omega} \left(\mathcal{L}_C^{sk^{\exists}(T)}\right)\text{-SA} + \mathrm{Clause}(\mathcal{L}_C^{sk^{\exists}(T)})\text{-DDEF} \\ + \forall_1(\mathcal{L}_C^{sk^{\exists}(T)})\text{-LDEF} + \mathcal{L}_C^{sk^{\exists}(T)}\text{-CSIND}.$$

By Lemma 5.2.7 we can eliminate induction split symbols. After that, we use Proposition 2.2.8 to deskolemize the induction schema. Therefore, the following theory is inconsistent

$$\mathcal{A}_0 + \mathrm{ACY} + sk^{\exists}(T) + sk^{\omega} \left(\mathcal{L}_C^{sk^{\exists}(T)}\right)\text{-SA} + \mathrm{Clause}(\mathcal{L}_C^{sk^{\exists}(T)})\text{-DDEF} \\ + \forall_1(\mathcal{L}_C^{sk^{\exists}(T)})\text{-LDEF} + \mathcal{L}_C^{sk^{\exists}(T)}\text{-CIND}.$$

By Lemma 5.2.9 we can reformulate $\mathcal{L}_C^{sk^{\exists}(T)}$-CIND as parameter-free structural induction on $\forall_1$ formulas. Furthermore, Lemma 5.2.11 allows us to eliminate split symbols from the induction and to absorb nullary Skolem symbols into parameterized structural induction. Hence, the following theory is inconsistent

$$\mathcal{A}_0 + sk^{\exists}(T) + sk^{\omega} \left(\mathcal{L}_C^{sk^{\exists}(T)}\right)\text{-SA} + \mathrm{Clause}(\mathcal{L}_C^{sk^{\exists}(T)})\text{-DDEF} \\ + \forall_1(\mathcal{L}_C^{sk^{\exists}(T)})\text{-LDEF} + \forall_1(\mathcal{L}(sk^{\exists}(T)) \cup \mathcal{L}_0)\text{-IND}. \qquad \square$$

In the following we will eliminate the remaining split symbols. The idea is to "unfold" the definitions by replacing the split symbols by their defining formulas. If the starting theory T is free of split symbols, then by the construction of the language $\mathcal{L}_C^T$, the unfolding process terminates and results in formulas that are free of split symbols. In particular, since the induction is already free of split symbols this process does not affect the induction.

Lemma 5.2.13 (Definition unfolding)**.** *Let T be a theory free of split symbols, then for every formula φ, there is a $\left(\mathcal{L}(\varphi) \setminus \left(\left(\mathcal{L}_C^T\right)_{\mathsf{D}} \cup \left(\mathcal{L}_C^T\right)_{\mathsf{L}}\right)\right) \cup \mathcal{L}(T_C^{\omega}(T))_{\overline{\mathsf{S}}}$ formula φ' such that*

$$\mathrm{Clause}(\mathcal{L}_C^T)\text{-DDEF} + \forall_1(\mathcal{L}_C^T)\text{-LDEF} \vdash \varphi \leftrightarrow \varphi'.$$

Proof. We have $\mathcal{L}_C^T = \mathcal{L}(T_C^{\omega}(T))_{\overline{\mathsf{I}}} = \bigcup_{i<\omega} \mathcal{L}(T_C^i(T))_{\overline{\mathsf{I}}}$. We proceed by induction on $i \in \mathbb{N}$ and show that for every $\mathcal{L}(T_C^i(T))_{\overline{\mathsf{I}}}$ formula $\varphi(\vec{x})$ there exists a $\mathcal{L}(T_C^i(T))_{\overline{\mathsf{S}}}$ formula $\varphi'(\vec{x})$ such that

$$\mathrm{Clause}(\mathcal{L}_C^T)\text{-DDEF} + \forall_1(\mathcal{L}_C^T)\text{-LDEF} \vdash \varphi \leftrightarrow \varphi'.$$

The base case $i = 0$ is trivial since T is free of split symbols and $T_C^0 = T$. For the induction step we assume the claim for i and we consider the language $\mathcal{L}(T_C^{i+1}(T))$. Let φ be a $\mathcal{L}(T_C^{i+1}(T))$ formula and let $\mathsf{D}_{\varphi_1}, \ldots, \mathsf{D}_{\varphi_n}, \mathsf{L}_{\psi_1}, \ldots, \mathsf{L}_{\psi_m}$ be all the definition and lemma split symbols occurring in φ. Then $\varphi_1, \ldots, \varphi_n$ and $\psi_1, \ldots, \psi_m$ are $\mathcal{L}(T_C^i(T))_{\overline{\mathsf{I}}}$

sentences. Hence, by the induction hypothesis there are $\mathcal{L}(T_C^i(T))_{\overline{\mathsf{S}}}$ sentences $\varphi'_1, \ldots, \varphi'_n, \psi'_1, \ldots, \psi'_m$ such that

$$\text{Clause}(\mathcal{L}_C^T)\text{-DDEF} + \forall_1(\mathcal{L}_C^T)\text{-LDEF} \vdash \bigwedge_{i=1}^{n} (\varphi_i \leftrightarrow \varphi'_i) \wedge \bigwedge_{i=1}^{m} (\psi_i \leftrightarrow \psi'_i)\,.$$

Hence, the formula $\varphi[\mathsf{D}_{\varphi_1}/\varphi'_1, \ldots, \mathsf{D}_{\varphi_n}/\varphi'_n, \mathsf{L}_{\psi_1}/\psi'_1, \ldots, \mathsf{L}_{\psi_n}/\psi'_m]$ is an $\mathcal{L}(T_C^i(T))_{\overline{\mathsf{S}}}$ formula. Therefore, we have

$$\text{Clause}(\mathcal{L}_C^T)\text{-DDEF} + \forall_1(\mathcal{L}_C^T)\text{-LDEF} \vdash \varphi \leftrightarrow \varphi[\mathsf{D}_{\varphi_1}/\varphi'_1, \ldots, \mathsf{D}_{\varphi_n}/\varphi'_n, \mathsf{L}_{\psi_1}/\psi'_1, \ldots, \mathsf{L}_{\psi_n}/\psi'_m].$$

Now let γ be a formula, then we expand all the occurrences of a split symbols D_φ and L_ψ with $\mathcal{L}_C^T$ sentences φ, ψ, by their split symbol free counterparts φ' and ψ' in order to obtain a formula γ'. It is clear that $\text{Clause}(\mathcal{L}_C^T)\text{-DDEF} + \forall_1(\mathcal{L}_C^T)\text{-LDEF} \vdash \gamma \leftrightarrow \gamma'$ and moreover $\mathcal{L}(\gamma) \subseteq \big(\mathcal{L}(\gamma) \setminus \big((\mathcal{L}_C^T)_{\mathsf{D}} \cup (\mathcal{L}_C^T)_{\mathsf{L}}\big)\big) \cup \mathcal{L}(T_C^\omega(T))_{\overline{\mathsf{S}}}$. □

There is a slight complication when unfolding definitions in the presence of Skolem axioms, because Skolem axioms are not closed under substitution of nullary predicate symbols by formulas. That is, we need to provide a suitable interpretation for Skolem symbols of formulas with split symbols. There is a trivial way to do this by making use of the axiom of choice as in the traditional model-theoretic proof of Proposition 2.2.11. However, in the proof of the following lemma we avoid using the axiom of choice.

Lemma 5.2.14. *Let T be a theory free of definition and lemma split symbols and M an $sk^\omega\big(\mathcal{L}(T_C^\omega(T))_{\overline{\mathsf{S}}}\big)$ structure such that $M \models sk^\omega\big(\mathcal{L}(T_C^\omega(T))_{\overline{\mathsf{S}}}\big)$-SA, then there exists an expansion M' of M such that*

$$M' \models sk^\omega\big(\mathcal{L}_C^T\big)\text{-SA} + \text{Clause}(\mathcal{L}_C^T)\text{-DDEF} + \forall_1(\mathcal{L}_C^T)\text{-LDEF}.$$

Proof. Let $M = (D, I)$, then we define the interpretation $I' \supseteq I$ as follows. Let $\mathsf{D}_\varphi, \mathsf{L}_\psi \in \mathcal{L}_C^T$, then we let $I'(\mathsf{D}_\varphi) = (\varphi')^M$ and $I'(\mathsf{L}_\psi) = (\psi')^M$, where φ' and ψ' are the sentences obtained by Lemma 5.2.13. It follows immediately that

$$(D, I') \models \text{Clause}(\mathcal{L}_C^T)\text{-DDEF} + \forall_1(\mathcal{L}_C^T)\text{-LDEF}. \qquad (*)$$

Furthermore, we proceed by induction on i and extend I' to an interpretation I'' by defining $I''(\mathfrak{s}_{(Qx)\varphi})$ for $\mathfrak{s}_{(Qx)\varphi} \in sk^i(\mathcal{L}_C^T) \setminus sk^\omega\big(\mathcal{L}(T_C^\omega(T))_{\overline{\mathsf{S}}}\big)$ such that $I'(\mathfrak{s}_{(Qx)\varphi}) = I(\mathfrak{s}_{(Qx)\varphi^*})$ for some $sk^\omega\big(\mathcal{L}(T_C^\omega(T))_{\overline{\mathsf{S}}}\big)$ formula φ^*. For the base case there is nothing to be done, since $sk^0(\mathcal{L}_C^T) = \mathcal{L}_C^T$. For the induction step, let $\mathfrak{s}_{(Qx)\varphi} \in sk^{i+1}(\mathcal{L}_C^T) \setminus sk^\omega\big(\mathcal{L}(T_C^\omega(T))_{\overline{\mathsf{S}}}\big)$. Then by the induction hypothesis there exists a $sk^\omega\big(\mathcal{L}(T_C^\omega(T))_{\overline{\mathsf{S}}}\big) \cup \big(\mathcal{L}_C^T\big)_{\mathsf{D}} \cup \big(\mathcal{L}_C^T\big)_{\mathsf{L}}$ formula ψ such that $M' \models \psi \leftrightarrow \varphi$. By Lemma 5.2.13 and by $(*)$ there exists a $sk^\omega\big(\mathcal{L}(T_C^\omega(T))_{\overline{\mathsf{S}}}\big)$ formula ψ' such that $M'' \models \psi \leftrightarrow \psi'$. Hence, we let $I''(\mathfrak{s}_{(Qx)\varphi}) = I(\mathfrak{s}_{(Qx)\psi'})$. It is now obvious that $(D, I'') \models sk^\omega\big(\mathcal{L}_C^T\big)$-SA. □

We are now ready to formulate the upper bound for Cruanes' method.

Proposition 5.2.15. *Let $\mathcal{S}$ be a sound saturation system, and T a theory free of split symbols. If $C_{\mathcal{S}}$ refutes the clause set $cnf(sk^{\exists}(T))$, then the theory*

$$\mathcal{A}_0 + T + sk^{\omega}\,(\mathcal{L}(T) \cup \mathcal{L}_0)\text{-SA} + \forall_1(\mathcal{L}(sk^{\exists}(T) \cup \mathcal{L}_0))\text{-IND}$$

is inconsistent.

Proof. Assume that $C_{\mathcal{S}}$ refutes $cnf(sk^{\exists}(T))$, then by Proposition 5.2.12 the following theory is inconsistent:

$$\begin{aligned}\mathcal{A}_0 + sk^{\exists}(T) + sk^{\omega}\left(\mathcal{L}_C^{sk^{\exists}(T)}\right)\text{-SA} + \text{Clause}(\mathcal{L}_C^{sk^{\exists}(T)})\text{-DDEF}\\ + \forall_1\left(\mathcal{L}_C^{sk^{\exists}(T)}\right)\text{-LDEF} + \forall_1(\mathcal{L}(sk^{\exists}(T)) \cup \mathcal{L}_0)\text{-IND}\end{aligned}$$

Hence, by using Lemma 5.2.14 to expand the split symbols by their defining sentences, the following theory is also inconsistent:

$$\mathcal{A}_0 + sk^{\exists}(T) + sk^{\omega}\left(\mathcal{L}(T_C^{\omega}(sk^{\exists}(T)))_{\overline{\mathsf{S}}}\right)\text{-SA} + \forall_1(\mathcal{L}(sk^{\exists}(T)) \cup \mathcal{L}_0)\text{-IND}.$$

Since $sk^{\omega}\left(\mathcal{L}(T_C^{\omega}(sk^{\exists}(T)))_{\overline{\mathsf{S}}}\right) = sk^{\omega}\left(\mathcal{L}(sk^{\exists}(T)) \cup \mathcal{L}_0\right) = sk^{\omega}\,(\mathcal{L}(T) \cup \mathcal{L}_0)$ and by Proposition 2.2.8, the claim follows. □

This upper bound can likely be improved by being more precise about the Skolem axioms that are actually needed. The situation is reminiscent of the one discussed in Question 3.2.7. In the case where the theory T in the proposition above is an $\exists_2$ theory we can even eliminate the Skolem axioms.

Corollary 5.2.16. *Let $\mathcal{S}$ be a sound saturation system, and T a Skolem-free, $\exists_2$ theory that is free of split symbols. If $C_{\mathcal{S}}$ refutes the clause set $cnf(sk^{\exists}(T))$, then the theory*

$$\mathcal{A}_0 + T + \forall_1(\mathcal{L}(T) \cup \mathcal{L}_0)\text{-IND}$$

is inconsistent.

Proof. If T is $\exists_2$, then $sk^{\exists}(T)$ extends the language of T by nullary Skolem symbols only. Hence, occurrences of these additional Skolem symbols in the induction formulas are absorbed by induction parameters. Now it suffices to apply Proposition 2.2.11. □

In Section 5.3 we will show that in the $\exists_2$ setting the upper bound of Corollary 5.2.16 can not be improved. As usual the upper bound of Corollary 5.2.16 gives us access to semantical arguments. In particular, this allows us to provide in Section 5.4 an unprovability result for the variant of Cruanes' calculus considered in this chapter.

5.3 A completeness result

In the previous section we have provided an upper bound for Cruanes' calculus. Moreover, we have observed that in the $\exists_2$ setting this upper bound is even Skolem-free. This upper

bound shows us that Cruanes' calculus is not stronger than induction for $\forall_1$ formulas. In this section we will show that in the $\exists_2$ setting the upper bound can not be improved.

The argument is essentially the same as the one given in Section 3.2.1. The crucial point of the argument is to provide a form of the compactness theorem that allows us to obtain induction formulas whose languages are compatible with the language generated by a sound saturation system extended by the rules given in Section 5.1. As in Section 3.2.1 we will provide a construction that makes use of proofs in the sequent calculus **G** in order to Skolemize induction formulas as needed. Thus, we reformulate the induction axioms so that the relevant strong quantifiers are prenexed and thus become easier to deal with.

Definition 5.3.1. *Let $\varphi(x, \vec{y}, \vec{z})$ be a formula, then the formula $J^C_{x,\vec{y}}\varphi$ is given by*

$$(\varphi(0, \vec{y}, \vec{z}) \wedge ((\forall \vec{y})\varphi(x, \vec{y}, \vec{z}) \rightarrow \varphi(s(x), \vec{y}, \vec{z}))) \rightarrow (\forall x)(\forall \vec{y})\varphi(x, \vec{y}, \vec{z}).$$

Lemma 5.3.2. *Let $\varphi(x, \vec{y})$ be a formula, then $\vdash (\exists x)(\exists \vec{y})J^C_x\varphi \leftrightarrow I_x(\forall \vec{y})\varphi$.*

Proof. Straightforward. □

In order to simplify the notation we introduce some abbreviation for the relevant induction contexts.

Definition 5.3.3. *Let $\varphi(x, \vec{y})$ be a quantifier-free formula, then the induction context $\mathcal{J}_x\varphi$ is given by*

$$\left(\neg \bigwedge_{j=1}^{n} C_j(x, \vec{y}), x, 0\right),$$

where $C_1(x, \vec{y}), \ldots, C_n(x, \vec{y})$ are $\mathcal{L}(\varphi)$ clauses such that

$$cnf(sk^{\exists}(\neg(\forall x)(\forall \vec{y})\varphi(x, \vec{y}))) = cnf(\neg\varphi(c, \vec{d})) = \{C_1(c, \vec{d}), \ldots, C_n(c, \vec{d})\}.$$

We can now formulate the adapted compactness that we will use in the completeness proof below.

Proposition 5.3.4 (Adapted compactness)**.** *Let T be a theory. If the theory*

$$\mathcal{A}_0 + T + \forall_1(\mathcal{L}(T) \cup \mathcal{L}_0)\text{-IND}$$

is inconsistent, then there is a finite set of formulas $S_0 \subseteq \mathcal{A}_0 \cup sk^{\exists}(T)$ and quantifier-free formulas $\varphi_1(x, \vec{y}), \ldots, \varphi_n(x, \vec{y})$ such that for $i = 1, \ldots, n$ with

$$S_i \coloneqq S_0 \cup \left\{(J^C_{x,\vec{y}}\varphi_j)[x/\mu_{\mathcal{J}_x\varphi_j}, \vec{y}/\nu_{\mathcal{J}_x\varphi_j}] \mid 1 \leq j < i\right\},$$

φ_{i-1} is an $\mathcal{L}(S_i)$ formula and S_n is inconsistent.

Before we prove Proposition 5.3.4 we prove the following helper lemma.

Lemma 5.3.5. *Let S be a finite set formulas not containing weak quantifiers such that $\mathcal{L}_0 \subseteq \mathcal{L}(S)$. Furthermore, let I be a finite set of formulas of the form $(\forall \vec{z})(\exists x)(\exists \vec{y})J^C_{x,\vec{y}}\varphi$*

with $\varphi(x, \vec{y}, \vec{z})$ a quantifier-free $\mathcal{L}(S)$ formula. If $S, I \Longrightarrow$ has a **G** *proof, then there are quantifier-free formulas $\psi_1(x, \vec{y}), \ldots, \psi_n(x, \vec{y})$ such that for $i = 1, \ldots, n$,*

$$\mathcal{L}(\psi_i) \subseteq \mathcal{L}(S) \cup \bigcup_{1 \leq j < i} \mathcal{L}(\psi_j[x/\mu_{\mathcal{J}_x\psi_j}, \vec{y}/\nu_{\mathcal{J}_x\psi_j}])$$

and $S + \{J^C_{x,\vec{y}}\psi_i[x/\mu_{\mathcal{J}_x\psi_i}, \vec{y}/\nu_{\mathcal{J}_x\psi_i}] \mid 1 \leq i \leq n\}$ is inconsistent.

Proof. If $S, I \Longrightarrow$ has a **G** proof, then it also has a **G** proof π in atomic cut normal form. We prove the claim by induction on the number of strong quantifier inferences of π. If π does not contain a strong quantifier inference, then by permuting quantifier inferences on ancestors of I downward we obtain a proof ρ of the sequent $S \Longrightarrow$. On the other hand, if π contains a strong quantifier inference, then since π is free of non-atomic cuts and S does not contain weak quantifiers, the principal formulas of strong quantifier inferences are ancestors of I. Hence, by permuting quantifier inferences downward we obtain a proof ρ of the form

$$\dfrac{\dfrac{\dfrac{\dfrac{(\rho'(x, \vec{y}))}{S, J^C_{x,\vec{y}}\varphi[\vec{z}/\vec{t}], I \Longrightarrow}}{S, (\exists \vec{y}) J^C_{x,\vec{y}}\varphi[\vec{z}/\vec{t}], I \Longrightarrow}\,\mathrm{L}\exists}{S, (\exists x)(\exists \vec{y}) J^C_{x,\vec{y}}\varphi[\vec{z}/\vec{t}], I \Longrightarrow}\,\mathrm{L}\exists}{S, (\forall \vec{z})(\exists x)(\exists \vec{y}) J^C_{x,\vec{y}}\varphi, I \Longrightarrow}\,\mathrm{L}\forall^*}{S, I \Longrightarrow ,}\,\mathrm{LC}$$

where $\mathrm{sqi}(\rho') < \mathrm{sqi}(\rho) \leq \mathrm{sqi}(\pi)$, $\varphi(x, \vec{y}, \vec{z})$ is a quantifier-free $\mathcal{L}(S)$ formula, $\vec{t}$ is a vector of ground $\mathcal{L}(S)$ terms. Now let $\psi := \varphi[\vec{z}/\vec{t}]$, then $\mathrm{sqi}(\rho'(\mu_{\mathcal{J}_{x,\vec{y}}\psi}, \nu_{\mathcal{J}_{x,\vec{y}}\psi})) = \mathrm{sqi}(\rho') < \mathrm{sqi}(\mu)$). Hence, we can apply the induction hypothesis to $\rho'(\mu_{\mathcal{I}}, \nu_{\mathcal{I}})$ to obtain the desired formulas. □

Proof of Proposition 5.3.4. If $\mathcal{A}_0 + T + \forall_1(\mathcal{L}(T) \cup \mathcal{L}_0)$-IND is inconsistent, then by Skolemizing T, applying Lemma 5.3.2 and the compactness theorem, there are finite sets of formulas $S'_0 \subseteq \mathcal{A}_0 + sk^{\exists}(T)$ and

$$I \subseteq \{(\forall \vec{z})(\exists x)(\exists \vec{y}) J_{x,\vec{y}}\varphi \mid \varphi(x, \vec{y}, \vec{z}) \text{ is a quantifier-free } \mathcal{L}(T) \cup \mathcal{L}_0 \text{ formula}\},$$

such that $S'_0 + I$ is inconsistent. If $\mathcal{L}(I) \not\subseteq \mathcal{L}(S'_0)$, then we obtain $S_0 \supseteq S'_0$ from S_0 by adding finitely many axioms of $\mathcal{A}_0 + sk^{\exists}(T)$ such that $\mathcal{L}(I) \subseteq \mathcal{L}(S_0)$. Now it suffices to apply Lemma 5.3.5 to S_0 and I in order to obtain the desired formulas. □

The proof of the completeness result given below makes use of the formulas obtained from Proposition 5.3.4 to construct a refutation consisting of two parts. In the first part we make use of the rules $\mathrm{LEM}^{\forall_1}_{\mathrm{S}}$ and $\mathrm{CIND}_{\mathrm{S}}$ to derive the clauses corresponding to the induction axioms. In particular, the lemma rule will be used before each application of the rule $\mathrm{CIND}_{\mathrm{S}}$ in order to ensure that there are suitable clauses on which we can carry

out induction. After that, the given refutationally complete saturation system system will be invoked to obtain a refutation of the clause set derived by the first part.

Theorem 5.3.6. *Let T be a theory and $\mathcal{S}$ a refutationally complete saturation system. If $\mathcal{A}_0 + T + \forall_1(\mathcal{L}(T) \cup \mathcal{L}_0)$-IND is inconsistent, then the saturation system $C_{\mathcal{S}}$ refutes $cnf(sk^{\exists}(T) \cup \mathcal{A}_0)$.*

Proof. Assume that $\mathcal{A}_0 + T + \forall_1$-IND is inconsistent, then obtain the sets $S_0, S_1, \ldots, S_n$ from Proposition 5.3.4. Now we proceed inductively and derive clause sets $\mathcal{C}_0, \ldots, \mathcal{C}_n$ such that $\mathcal{L}(S_i) \subseteq \mathcal{L}(\mathcal{C}_i)$ and $\mathcal{C}_i \models S_i$. We let $\mathcal{C}_0 = cnf(S_0)$, then the condition above is met. Now assume that we have obtained $\mathcal{C}_i$, then we construct the clause set $\mathcal{C}_{i+1}$ as follows. Let $S_{i+1} = S_i \cup \{J_{x,\vec{y}}\varphi_{i+1}[x/\mu_{\mathcal{J}_x\varphi_{i+1}}, \vec{y}/\nu_{\mathcal{J}_x\varphi_{i+1}}]\}$. Since $\mathcal{L}(\varphi_{i+1}) \subseteq \mathcal{L}(S_i) \subseteq \mathcal{L}(\mathcal{C}_i)$, we can now derive the clause set $\mathcal{C}'_{i+1}$ by an application of $\mathrm{LEM}_{\mathrm{S}}^{\forall_1}$ to the sentence $(\forall x)(\forall \vec{y})\varphi_{i+1}$. Let $cnf(sk^{\exists}(\neg(\forall x)(\forall \vec{y})\varphi_{i+1})) = cnf(\varphi_{i+1}(c, \vec{d})) = \{C_1(c, \vec{d}), \ldots, C_n(c, \vec{d})\}$, where $C_1(x, \vec{y}), \ldots, C_n(x, \vec{y})$ are $\mathcal{L}(\varphi_{i+1})$. Then,

$$\{C_1(c, \vec{d}) \vee \mathsf{L}_{(\forall x)(\forall \vec{y})\varphi_{i+1}}, \ldots, C_n(c, \vec{d}) \vee \mathsf{L}_{(\forall x)(\forall \vec{y})\varphi_{i+1}}\} \subseteq \mathcal{C}'_{i+1}.$$

Now we may apply the induction rule $\mathrm{CIND}_{\mathrm{S}}$ to $\mathcal{C}'_{i+1}$ with the induction context

$$\mathcal{I} \coloneqq \mathcal{J}_{x,\vec{y}}\varphi_{i+1} = \left(\neg \bigwedge_{i=1}^{n} C_i(x, \vec{y}), x, 0\right)$$

in order to obtain the clause set $\mathcal{C}_{i+1}$ given by

$$\mathcal{C}'_{i+1} \cup cnf(IPS_{\mathcal{I}}) \cup cnf(IPD_{\mathcal{I}} \vee \mathsf{L}_{(\forall x)(\forall \vec{y})\varphi_{i+1}}). \tag{$*$}$$

The symbol $(\mu_{\mathcal{J}_x\varphi_{i+1}})_0$ and the symbols in $\nu_{\mathcal{J}_x\varphi_{i+1}}$ occur in $\mathcal{C}_{i+1}$ if and only if they occur in S_{i+1}. Hence $\mathcal{L}(S_i) \subseteq \mathcal{L}(C_{i+1})$. It remains to show that

$$\mathcal{C}_{i+1} \models J^C_{x,\vec{y}}\varphi_{i+1}[x/\mu_{\mathcal{J}_x\varphi_{i+1}}, \vec{y}/\nu_{\mathcal{J}_x\varphi_{i+1}}].$$

First of all observe that $\bigwedge_{i=1}^{n} C_i(c, \vec{d})$ is logically equivalent to $\neg\varphi_{i+1}(c, \vec{d})$. Because c and $\vec{d}$ are Skolem constants of prenex quantifiers, these symbols do not occur in $\varphi_{i+1}(x, \vec{y})$ and therefore also do not occur in $C_1(x, \vec{y}), \ldots, C_n(x, \vec{y})$. Hence, $\neg\varphi_{i+1}(x, \vec{y})$ and $\bigwedge_{i=1}^{n} C_i(x, \vec{y})$ are logically equivalent. Thus $\neg \bigwedge_{i=1}^{n} C_i(x, \vec{y})$ is logically equivalent to $\varphi_{i+1}(x, \vec{y})$.

Now let M be a structure such that $M \models \mathcal{C}_{i+1}$ and assume that

$$M \models \varphi_{i+1}(0, \nu_{\mathcal{I}}(\vec{y})),$$
$$M \models (\forall \vec{y})\varphi_{i+1}(\mu_{\mathcal{I}}, \vec{y}) \to \varphi_{i+1}(s(\mu_{\mathcal{I}}), \nu_{\mathcal{I}}).$$

If $M \models \mathsf{L}_{(\forall x)(\forall \vec{y})\varphi_{i+1}}$, then since $cnf((\forall x)(\forall \vec{y})\varphi_{i+1}) \leftarrow \mathsf{L}_{(\forall x)(\forall \vec{y})\varphi_{i+1}} \subseteq \mathcal{C}_{i+1}$, we have $M \models (\forall x)(\forall \vec{y})\varphi_{i+1}$ and we are done. Otherwise, by $(*)$ we have $M \models IPD_{\mathcal{I}}$ and $M \models IPS_{\mathcal{I}}$. Hence, $M \models \mathsf{I}^{x,0}_{\mathcal{I}} \vee \mathsf{I}^{x,s(x)}_{\mathcal{I}}$. If $M \models \mathsf{I}^{x,0}_{\mathcal{I}}$, then we have $M \models \neg IP^{x,0}_{\mathcal{I}}[x/\mu_{\mathcal{I}}, \vec{y}/\nu_{\mathcal{I}}]$.

Since, $\vdash IP^{x,0}_{\mathcal{I}} \leftrightarrow \varphi_{i+1}(0,\vec{y})$, this case is impossible. On the other hand, if $M \models \mathrm{I}^{x,s(x)}_{\mathcal{I}}$, then we have

$$M \models \neg IP^{x,s(x)}_{\mathcal{I}}[x/\mu_{\mathcal{I}}, \vec{y}/\nu_{\mathcal{I}}].$$

Hence, since $\vdash IP^{x,s(x)}_{\mathcal{I}} \leftrightarrow ((\forall \vec{y})\varphi_{i+1}(x,\vec{y}) \rightarrow \varphi_{i+1}(x,\vec{y}))$, this contradicts the assumption. Thus $M \models J^{C}_{x,\vec{y}}[x/\mu_{\mathcal{I}}, \vec{y}/\nu_{\mathcal{I}}]$.

Since S_n is inconsistent, also $\mathcal{C}_n$ is inconsistent and by the refutational completeness of $\mathcal{S}$ we obtain a refutation of $\mathcal{C}_n$. Hence we have obtained a $C_{\mathcal{S}}$ refutation of $cnf(sk^{\exists}(T) \cup \mathcal{A}_0)$. □

In the $\exists_2$ setting we have the following characterization of refutation by Cruanes' system.

Corollary 5.3.7. *Let $\mathcal{S}$ be a sound and refutationally complete saturation system and T a Skolem-free, $\exists_2$ theory that is free of split symbols, then the saturation system $C_{\mathcal{S}}$ refutes $cnf(\mathcal{A}_0 + sk^{\exists}(T))$ if and only if $\mathcal{A}_0 + T + \forall_1(\mathcal{L}(T) \cup \mathcal{L}_0)$-IND is inconsistent.*

Proof. An immediate consequence of Corollary 5.2.16 and Theorem 5.3.6. □

This shows that Cruanes' system is in the $\exists_2$ setting in principle at least as strong as the systems based on the various rules considered in Chapter 4. However, the result relies crucially on applications of the lemma rule $\mathrm{LEM}^{\forall_1}_{\mathrm{S}}$ in order to introduce the necessary syntactic material. A practical implementations typically uses the lemma rule heuristically and therefore does not guarantee completeness in the sense of Corollary 5.3.7.

However, what this result and its proof tell us is that we could replace the induction rule $\mathrm{CIND}_{\mathrm{S}}$ by a simpler rule that operates on ground subclauses and relies on the induction axioms $I_x\varphi$, where φ is a $\forall_1$ formula. In particular, simultaneous induction and big-step induction can be simulated by this simplified induction rule together with the lemma rule $\mathrm{LEM}^{\forall_1}_{\mathrm{S}}$. Similarly, other improvements that do not strengthen the system beyond induction for $\forall_1$ induction can be reduced to the simplified induction rule. This architectural remark is to a certain extent of practical value as the induction rule has to be implemented only once and heuristics can be included afterwards by suitable applications of $\mathrm{LEM}^{\forall_1}_{\mathrm{S}}$. This has the advantage of reducing the soundness of the implementation of a heuristic to the soundness of the implementation of the induction rule. This observation gives rise to the interesting question whether such a simplification is possible in a setting with more complicated datatypes.

Remark 5.3.8. *The discussion of the paragraph above is reminiscent of Remark 3.1.12 where we have seen that in presence of sufficiently many Skolem axioms it even suffices to use induction over quantifier-free formulas. For the calculus considered in this chapter such a simplification of the induction rule is not possible, despite the Skolem axioms introduced by the rule $\mathrm{LEM}^{\forall_1}_{\mathrm{S}}$. This is because the lemma rule $\mathrm{LEM}^{\forall_1}_{\mathrm{S}}$ introduces Skolem axioms for sentences only.*

5.4 Unprovability

In Section 5.2 we have derived an upper bound on the strength of the variant for Cruanes' calculus given in Section 5.1. In this section we will make use of this upper bound derived to provide a simple unprovability result for this calculus. In particular, the unprovability result that we provide here separates $\exists_1$ induction and $\forall_1$ induction. Such an unprovability result will be become particularly interesting in Chapter 6 where we consider systems that are based on induction over $\exists_1$ formulas.

Over the base theory of linear arithmetic B induction for $\exists_1$ formulas coincides with induction over $\forall_1$ formulas. Hence, we need to work in a different setting in order to find such a sentence that separates induction for $\forall_1$ formulas and induction for $\exists_1$ formulas. Moreover, $B + \exists_1(\mathcal{L}(T))$-IND is even a complete theory, thus we will not find any unprovability results in this setting. As in we did in Chapter 4 we will work in a very simple setting that provides besides the symbols $0/0$ and $s/1$ a binary predicate symbol whose intended interpretation is the graph of the function $n \mapsto 2n$.

Definition 5.4.1. *Let $D_2/2$ be a predicate symbol. The theory $\mathcal{A}_7$ extends the theory $\mathcal{A}_0$ by the universal closure of the formulas*

$$D_2(0,0), \tag{A7.1}$$
$$D_2(x,y) \to D_2(s(x), s(s(y))), \tag{A7.2}$$
$$D_2(x,y_1) \wedge D_2(x,y_2) \to y_1 = y_2. \tag{A7.3}$$

We proceed as usual by constructing a suitable $\mathcal{L}(\mathcal{A}_7)$ structure in which the predicate $D_2(x,y)$ is not total in y.

Definition 5.4.2. *Let the domain of the $\mathcal{L}(\mathcal{A}_7)$ structure M consist of pairs $(i,n) \in \{0,1\} \times \mathbb{Z}$ such that $i = 0$ implies $n \in \mathbb{N}$ and let M interpret the symbols of $\mathcal{L}(\mathcal{A}_7)$ as follows*

$$0^M = (0,0),$$
$$s^M((i,n)) = (i, n+1),$$
$$D_2^M = \{((0,n),(0,2n)) \mid n \in \mathbb{N}\}.$$

Lemma 5.4.3. $M \models \mathcal{A}_7 + (\forall x)(\exists y)[x = 0 \vee x = s(y)]$.

Proof. It is obvious that $M \models 0 \neq s(x)$ and $M \models s(x) = s(y) \to x = y$. Furthermore, it is obvious that $M \models (\exists y)(x = 0 \vee x = s(y))$. Clearly $((0,0),(0,0)) \in D_2^M$, hence $M \models$ (A7.1). Let $((i_1,n_1),(i_2,n_2)) \in D_2^M$, then $i_1 = i_2 = 0$ and $n_2 = 2n_1$. Hence, $n_2 + 2 = 2(n_1 + 1)$, thus $((0,n_1+1),(0,n_2+2)) \in D_2^M$. Therefore, $M \models$ (A7.2). Finally, by the definition of M it is obvious that $M \models$ (A7.3). □

In the following we will show that M satisfies the induction schema for quantifier-free $\mathcal{L}(\mathcal{A}_7)$ formulas. In order to simplify the considerations, we will mainly work with $\forall_1$ formulas whose quantifier-free matrix is in conjunctive normal form. Formulas in this representation can easily be simplified by eliminating certain literals.

Lemma 5.4.4. *Let $C(\vec{x},\vec{y},\vec{z})$ be a clause, then there exists $N \in \mathbb{N}$ such that over $\mathcal{A}_0$ the formula $(\forall\vec{y})C(s^N(\vec{x}),\vec{y},\vec{z})$ is equivalent to $(\forall\vec{y})C'(\vec{x},\vec{y},\vec{z})$ where C' is a clause not containing literals of the form $s^{k_1}(x_i) \neq s^{k_2}(y_j)$.*

Proof. We proceed by induction on the number of variables of $\vec{y}$ occurring in C. If C contains a literal of the form $s^{k_1}(x_i) \neq s^{k_2}(y_j)$, then $(\forall\vec{y})C$ is equivalent over $\varnothing$ to $(\forall\vec{y})\big(s^{k_1}(x_i) = s^{k_2}(y_j) \to C\big)$. Then by letting $N = k_2$, $(\forall\vec{y})C(s^N(\vec{x}),\vec{y},\vec{z})$ is equivalent over $\mathcal{A}_0$ to $(\forall\vec{y})\big(s^{k_1}(x_i) = y_j \to C(s^N(\vec{x}),\vec{y},\vec{z})\big)$. Let $C' \coloneqq C(s^N(\vec{x}),\vec{y},\vec{z})$, then $(\forall\vec{y})C(s^N(\vec{x}),\vec{y},\vec{z})$ is equivalent over $\mathcal{A}_0$ to the formula $(\forall\vec{y})C'[y_j/s^{k_i}(x_i)]$. Since $C'[y_j/s^{k_i}(x_i)]$ does not contain the variable y_j we may apply the induction hypothesis to $(\forall\vec{y})C'[y_j/s^{k_i}(x_i)]$ in order to obtain N' and a disjunction of literals C'' such that $(\forall\vec{y})C'[y_j/s^{k_i}(x_i)][\vec{x}/s^{N'}(\vec{x})]$ is equivalent over $\mathcal{A}_0$ to $(\forall\vec{y})C''$. Hence, $(\forall\vec{y})C[\vec{x}/s^{N+N'}(\vec{x})]$ is equivalent over $\mathcal{A}_0$ to $(\forall\vec{y})C'$ and C' does not contain a negative literal of the form $s^{k_1}(x_i) \neq s^{k_2}(y_j)$. □

The following lemma shows that atoms of $\mathcal{L}(\mathcal{A}_7)$ behave in M very similar to equations between linear polynomials.

Lemma 5.4.5. *Let $\theta(x,\vec{y})$ be a an $\mathcal{L}(\mathcal{A}_7)$ atom and $\vec{b}$ a finite sequence of elements of M. If $M \not\models \theta(x,\vec{b})$, then there exists $N \in \mathbb{N}$ such that for $n \geq N$*

$$M \not\models \theta((1,-n),\vec{b}) \text{ and } M \not\models \theta(\overline{n},\vec{b}).$$

Proof. If θ does not contain x, then we are done by letting $N = 0$. Now assume that θ is an equational atom. Then θ is without loss of generality of the form $s^{k_1}(x) = t_2$. If t_2 contains x, then $t_2 = s^{k_2}(x)$ with $k_1 \neq k_2$. Then we are done by letting $N = 0$. If t_2 does not contain x, then let $M \models t_2(\vec{b}) = (i,m)$. Let $n \geq m+1+k_1$, then $n + k_1 \geq m + 1 + 2k_1 > m$. Similarly, $-n + k_1 \leq -(m+1) < m$. Now assume that θ is an atom of the form $D_2(t_1(x,\vec{y}),t_2(x,\vec{y}))$. By the definition of D_2^M we already have $M \not\models D_2(t_1((1,n),\vec{b}),t_2((1,n),\vec{b}))$ for all $n \in \mathbb{Z}$. Suppose that x occurs in t_1, then t_1 is of the form $s^{k_1}(x)$ If x occurs in t_2, then t_2 is of the form $s^{k_2}(x)$. Let $n \geq k_2 + 1$, then $2n + 2k_1 \geq n + k_2 + 1 > n + k_2$. If x does not occur in t_2, then let $M \models t_2(\vec{b}) = (i,m)$. If $i = 1$, then we are done by letting $N = 0$. Otherwise, let $n \geq m+1$, then we have $2n + 2k_1 \geq n + m + 1 > m$. If x does not occur in t_1, but occurs in t_2 then t_2 is of the form $s^k(x)$ and let $M \models t_1(\vec{b}) = (i,m)$. If $i = 1$, then we are done by letting $N = 0$. Otherwise, let $n \geq 2m+1$, then $2m < 2m + 1 + k$. □

The following lemma extends the observation of the previous lemma to quantifier-free $\mathcal{L}(\mathcal{A}_7)$ formulas.

Lemma 5.4.6. *Let $\varphi(x,\vec{y})$ be a quantifier-free formula and $\vec{b}$ a finite sequence of elements of M, then there exists $N \in \mathbb{N}$ such that either $M \models \varphi((1,-n),\vec{b})$ and $M \models \varphi(\overline{n},\vec{b})$ for all $n \geq N$ or $M \not\models \varphi((1,-n),\vec{b})$ and $M \not\models \varphi(\overline{n},\vec{b})$ for all $n \geq N$.*

Proof. Let $\theta_1(x,\vec{y}), \ldots, \theta_m(x,\vec{y})$ be all the atoms of φ such that $M \not\models \theta_i(x,\vec{b})$ for $i = 1,\ldots,m$. Then by Lemma 5.4.5 there exists $N \in \mathbb{N}$ such that for all $n \geq N$, $M \not\models \theta_i((1,-n),\vec{b})$ and $M \not\models \theta_i(\overline{n},\vec{b})$ for $i = 1,\ldots,m$. Thus we clearly have $M \models \varphi((1,-n),\vec{b})$ and $M \models \varphi(\overline{n},\vec{b})$ for all $n \geq N$ or $M \not\models \varphi((1,-n),\vec{b})$ and $M \not\models \varphi(\overline{n},\vec{b})$. □

By using the previous results it is now straightforward to show that M satisfies induction for $\forall_1(\mathcal{L}(\mathcal{A}_7))$ formulas.

Proposition 5.4.7. $M \models \forall_1(\mathcal{L}(\mathcal{A}_7))$-IND.

Proof. Let $\varphi(x, \vec{z})$ be a $\forall_1(\mathcal{L}(\mathcal{A}_7))$ formula and $\vec{c}$ a finite sequence of elements of M. Now assume

$$M \models \varphi(0, \vec{c}), \tag{$*$}$$
$$M \models \varphi(x, \vec{c}) \to \varphi(s(x), \vec{c}). \tag{$\star$}$$

By obtaining a logically equivalent conjunctive normal form for φ, moving quantifiers inwards, applying Lemma 5.4.4 to the conjuncts and rearranging the disjuncts, we obtain an $N \in \mathbb{N}$ such that

$$\mathcal{A}_0 \vdash \varphi(s^N(x), \vec{z}) \leftrightarrow \bigwedge_{i=1}^{m} \left(D_i \vee (\forall \vec{y})(C_i^+ \vee C_i^-)\right),$$

where, for $i = 1, \ldots, m$, $D_i(x, \vec{z})$, $C_i^-(x, \vec{y}, \vec{z})$, $C_i^+(x, \vec{y}, \vec{z})$ are disjunctions of literals such that C_i^- contains negative literals only and no literals of the form $s^{k_1}(x) \neq s^{k_2}(y_i)$, C_i^+ contains positive literals only, and every literal in $C_i^- \vee C_i^+$ contains a variable of $\vec{y}$. For the sake of legibility we let $C_i \coloneqq C_i^- \vee C_i^+$.

We have to show $M \models \varphi(x, \vec{c})$. By a straightforward induction and making use of $(*)$ and $(\star)$ we obtain $M \models \varphi(\overline{n}, \vec{c})$ for all $n \in \mathbb{N}$. Since $M \models (\exists y)[x = 0 \vee x = s(y)]$, it suffices to show $M \models (\forall x)\varphi(s^N(x), \vec{c})$. Let $a \in \mathcal{D}(M)$, if $a = (0, n)$, then $M \models a = \overline{n}$ and we are done. Let us now consider the case where $a = (1, n)$ with $n \in \mathbb{Z}$. By the above it thus suffices to show

$$M \models \bigwedge_{i=1}^{m} \left(D_i(a, \vec{c}) \wedge (\forall \vec{y}) C_i(a, \vec{y}, \vec{c})\right). \tag{$\dagger$}$$

By Lemma 5.4.6 there exists $R \in \mathbb{N}$ with $R \geq |n|$ such that for $i \in \{1, \ldots, m\}$, either $M \models D_i((1, -r), \vec{c})$ and $M \models D_i(\overline{r}, \vec{c})$ for $r \geq R$ or $M \not\models D_i((1, -r), \vec{c})$ and $M \not\models D_i(\overline{r}, \vec{c})$ for $r \geq R$. Now suppose that

$$M \models \bigwedge_{i=1}^{m} \left(D_i((1, -R), \vec{c}) \vee (\forall \vec{y}) C_i((1, -R), \vec{y}, \vec{c})\right), \tag{$\ddagger$}$$

then, since $-R \leq n$, a straightforward induction making use of $(\star)$ gives $(\dagger)$. Hence, it suffices to show $(\ddagger)$. We proceed indirectly and assume that there is $i_0 \in \{1, \ldots, m\}$ and a finite sequence $\vec{b}$ of elements of M such that

$$M \not\models D_{i_0}((1, -R), \vec{c}) \vee C_{i_0}((1, -R), \vec{b}, \vec{c}). \tag{$\diamond$}$$

Let us now first show that $C_{i_0}^-$ does not contain the variable x. Suppose that the variable x occurs in an equational literal of $C_{i_0}^-$. Then this literal is of the form $s^{k_1}(x) \neq$

$s^{k_2}(y_i)$, but such literals do not occur in $C^-_{i_0}$. Now suppose that x occurs in a literal of the form $\neg D_2(t_1, t_2)$. There are two cases to consider. If x occurs in t_1, then $t_1 = s^k(x)$ and by the definition of D_2^M we have $M \models \neg D_2((1, -R + k), t_2(\vec{b}))$, which contradicts $(\diamond)$. If x occurs in t_2, then we proceed analogously.

Since in particular $M \not\models C^+_{i_0}(x, \vec{b}, \vec{c})$, we can apply Lemma 5.4.5 to the atoms in $C^+_{i_0}$ in order to obtain $R' \geq R$ such that $M \not\models C^+_{i_0}((1, -R'), \vec{b}, \vec{c})$ and $M \not\models C^+_{i_0}(\overline{R'}, \vec{b}, \vec{c})$. Moreover, since $R' \geq R$ and $M \not\models D_{i_0}((1, -R), \vec{c})$, we have $M \not\models D_{i_0}((1, R'), \vec{c})$ and $M \not\models D_{i_0}(\overline{R'}, \vec{c})$. Since $C^-_{i_0}$ does not contain x we furthermore have $M \not\models C^-_{i_0}((1, R'), \vec{b}, \vec{c})$ and $M \not\models C^-_{i_0}(\overline{R'}, \vec{b}, \vec{c})$. Thus, $M \not\models D_{i_0}(\overline{R'}, \vec{c}) \vee (\forall \vec{y}) C_{i_0}(\overline{R'}, \vec{c})$. Hence, $M \not\models \varphi(\overline{N + R'}, \vec{c})$. Contradiction! □

Since $D_2(x, y)$ is not total in M in its second argument we now obtain the desired independence result.

Theorem 5.4.8. $\mathcal{A}_7 + \forall_1(\mathcal{L}(\mathcal{A}_7))$-IND $\not\vdash (\forall x)(\exists y) D_2(x, y)$.

Proof. Observe that $M \not\models D_2((1, 0), d)$ for all $d \in \mathcal{D}(M)$. □

This readily gives us an unprovability result for Cruanes' system.

Corollary 5.4.9. *Let $\mathcal{S}$ be a sound saturation system, then $C_{\mathcal{S}}$ does not refute the clause set $cnf(sk^{\exists}(\mathcal{A}_7 + (\exists x)(\forall y) \neg D_2(x, y)))$*

Proof. Assume that the clause set is refuted by $C_{\mathcal{S}}$, then by the upper bound for Cruanes system (Corollary 5.2.16) the sentence $(\forall x)(\exists y) D_2(x, y)$ is provable in the theory $\mathcal{A}_7 + \forall_1(\mathcal{L}(\mathcal{A}_7))$-IND. This contradicts Theorem 5.4.8. □

The independence result of Theorem 5.4.8 is slightly more involved than the ones considered in Chapter 4 because the sentence $(\forall x)(\exists y) D_2(x, y)$ contains a quantifier alternation. Totality statements such as the one above are interesting for automated inductive provers because the totality and well-definedness of a predicate justifies a functional extension. Especially in theory exploration systems and proof assistants, carrying out such an extension could be useful because it may be more convenient to work with a function symbol rather than a defining formula. However, in current practice, automated inductive theorem provers are mostly used to prove quantifier-free statements.

By the upper bounds given by Chapter 4 it is straightforward to see that the unprovability result given above also applies to the variants of the single-clause and multi-clause induction rules discussed in Chapter 4. In the next chapter we consider an induction mechanism in which the clause set of Corollary 5.4.9 has a natural refutation. Moreover, we will see that an unprovability results such as the one above, that can be overcome by induction on existentially quantified formulas, are only possible over very simple background theories that do not prove some simple facts about cut-off subtraction, see Lemma 5.4.10. In a slightly richer setting the induction schema for $\forall_1$ formulas is already as powerful as the induction schema for $\exists_1$ formulas

Lemma 5.4.10. *Let T be a theory and let "$x \dot{-} y = z$" denote an $\exists_0(\mathcal{L}(T))$ formula with free variables x, y, and z such that T proves*

$$\text{“}x \dot{-} 0 = z\text{”} \leftrightarrow z = x,$$
$$\text{“}x \dot{-} s(y) = z\text{”} \leftrightarrow (\exists v)\big(\text{“}x \dot{-} y = v\text{”} \wedge ((v = 0 \wedge z = 0) \vee v = s(z))\big),$$
$$\text{“}x \dot{-} x = 0\text{”}.$$

Then $T + \forall_n(\mathcal{L}(T))$-IND $\equiv T + \exists_n(\mathcal{L}(T))$-IND, for all $n \in \mathbb{N}$.

Proof. We show the direction from right to left. The other direction is analogous. Let $\psi(x, \vec{z})$ be an $\exists_n(\mathcal{L}(T))$ formula and assume $\psi(0, \vec{z})$ and $(\forall x)(\psi(x, \vec{z}) \to \psi(s(x), \vec{z}))$. We proceed indirectly and assume that $\neg\psi(x_0, \vec{z})$ for some x_0. We proceed by induction x in the formula $(\forall y)\big(\text{“}x_0 \dot{-} x = y\text{”} \to \neg\psi(y, \vec{z})\big)$. Clearly, this formula is logically equivalent to a $\forall_n(\mathcal{L})$ formula. For the induction base, assume "$x_0 \dot{-} 0 = y$". Then we have $y = x_0$, thus, by the assumption we have $\neg\psi(x_0, \vec{z})$.

For the induction step assume $(\forall y)\big(\text{“}x_0 \dot{-} x = y\text{”} \to \neg\psi(y, \vec{z})\big)$ and "$x_0 \dot{-} s(x) = y_0$". Hence, there exists v such that "$x_0 \dot{-} x = v$ and $(v = 0 \wedge y_0 = 0) \vee y_0 = s(v)$. If $v = 0$ and $x = 0$, then by the induction hypothesis $\neg\psi(v, \vec{z})$, that is $\neg\psi(0, \vec{z})$ which contradicts the assumptions. If $v = s(y_0)$, then by the induction hypothesis we first obtain $\neg\psi(v, \vec{z})$, that is, $\neg\psi(s(y_0), \vec{z})$, hence by the assumption we obtain $\neg\psi(y_0, \vec{z})$. Thus we have $(\forall x)(\forall y)(\text{“}x_0 \dot{-} x = y\text{”} \to \neg\psi(y, \vec{z}))$. Hence, "$x_0 \dot{-} x_0 = 0$" $\to \neg\psi(0, \vec{z})$. Since "$x_0 \dot{-} x_0 = 0$", we obtain $\neg\psi(0, \vec{z})$. Contradiction! □

By Herbrand's theorem (see Theorem 2.1.5) one can show that an analogous property does not hold in general for the induction rules. The proof above relies on the introduction of an additional induction parameter. In general parameter-free induction over $\exists_1$ and $\forall_1$ formulas are also not the same.

In the same way as we have shown the independence result of Theorem 5.4.8 we could obtain similar results for predicates that describe the graphs of polynomials and other functions. It could be interesting to investigate whether this idea can be systematized based on the growth rate of the function under consideration. For example, a positive answer to the following question would by Herbrand's theorem (see Theorem 2.1.5) immediately yield an alternative proof of Theorem 5.4.8.

Question 5.4.11. *Do we have $\mathcal{A}_7 + \forall_1(\mathcal{L}(\mathcal{A}_7))$-IND $\equiv_{\exists_1(\mathcal{L}(\mathcal{A}_7))}$ $\mathcal{A}_7 + \forall_1(\mathcal{L}(\mathcal{A}_7))$-IND^R?*

6 Clause set cycles

In this chapter we will consider another integration of induction into saturation-based theorem provers. This extension of saturation provers is based on the detection of cyclic dependencies between the clause sets that are derived by the prover. This extension differs from the methods considered in Chapters 3 and 4 in that the mechanism does not derive additional clauses and instead terminates the refutation once a cyclic dependency is detected. The cyclic dependencies that we consider in this chapter are an abstraction of the cycles detected by Kersani and Peltier's n-clause calculus [KP13; Ker14]—an extension of the superposition calculus by a cycle detection mechanism.

The content of this chapter is mostly based on the articles [HV20] and [HV22]. However, this chapter also provides some new complementary results. In Section 6.1 we introduce the notion of clause set cycle and discuss how we consider that this formalism relates to saturation-based provers. Moreover, we show that some practically motivated extensions of clause set cycles do not result in a stronger formalism. In Section 6.2 we compare clause set cycles to other induction systems. In particular, we will see that refutation by a clause set cycle naturally overcomes the unprovability example for the variant of Cruanes' calculus considered in Chapter 5. After that, we give in Section 6.3 a characterization of refutation by a clause set cycle in terms of a logical theory by discerning the logical features of clause set cycles. Finally, in Section 6.4 we provide several unprovability results for refutation by a clause set cycle. These unprovability results exploit different logical features of clause set cycles and therefore motivate various extensions of AITP systems. Some of these extension are not straightforward to formulate as extensions of clause set cycles and therefore make the logical characterization of Section 6.3 particularly useful.

6.1 Clause set cycles

Refutation by a clause set cycle is a formalism introduced in [HV20] by the authors of this article to describe abstractly the inductive arguments that take place in the n-clause calculus [KP13; Ker14]. The n-clause calculus is an extension of the superposition calculus by a mechanism that detects cyclic dependencies between the derived clauses. These cyclic dependencies correspond to arguments by infinite descent and thus establish the inductive unsatisfiability of a set of clauses. The notion of refutation by a clause set cycle abstracts the underlying superposition calculus and the detection of the cycle in that proof system and therefore extracts the essence of the arguments by infinite descent that may appear in refutations by the n-clause calculus.

Since all the variables occurring in clauses are implicitly universally quantified, a clause set does not have a free variable on which we can carry out an argument by induction.

Instead we will rely on a special constant symbol η, on which arguments by infinite descent will take place. This is in analogy to the special constant n that is used by the n-clause calculus for the same purpose, see [KP13]. The constant η can be thought of as a Skolem constant, that is selected before a refutation is attempted. In particular, clauses may of course contain other Skolem symbols besides η.

Carrying out arguments by infinite descent (or induction) only on positions of constants is unsurprisingly very restricting (see Corollary 3.1.11). Since clause set cycles are used as an abstraction of the inductive cycles of the n-clause calculus, we did not extend the formalism to allow arguments to take place in more varied positions. The logical characterization that we give in Section 6.3 makes considering such extensions easier. In particular, of the unprovability result given in this chapter, only Corollary 6.4.4, does not rely on this restriction. A method that lifts this restriction has been proposed in [EP20].

Remark 6.1.1. *In the literature [KP13; Ker14; HV20] a constant such as η is usually called a parameter. In order to avoid confusion with induction parameters in the sense of Definition 2.5.9 we will not use this designation.*

Let $\mathcal{C}$ be a clause set possibly containing η, then we write $\mathcal{C}(\eta)$ to indicate all the occurrences of η in $\mathcal{C}$. Let furthermore t be a term, then $\mathcal{C}(t)$ denotes the clause set obtained by replacing all the occurrences of η in $\mathcal{C}$ by t.

Definition 6.1.2 (Refutation by a clause set cycle)**.** *Let $\mathcal{L}$ be a first-order language not containing η. A finite $\mathcal{L} \cup \{\eta\}$ clause set $\mathcal{C}(\eta)$ is called an $\mathcal{L}$ clause set cycle if it satisfies the following conditions*

$$\mathcal{C}(s(\eta)) \models \mathcal{C}(\eta), \tag{C1}$$
$$\mathcal{C}(0) \models \bot. \tag{C2}$$

Let $\mathcal{D}(\eta)$ be an $\mathcal{L} \cup \{\eta\}$ clause set, then $\mathcal{D}(\eta)$ is refuted by an $\mathcal{L}$ clause set cycle $\mathcal{C}(\eta)$ if

$$\mathcal{D}(\eta) \models \mathcal{C}(\eta). \tag{C3}$$

A clause set cycle represents an argument by infinite descent in the following sense. Suppose there is an $L \cup \{\eta\}$ structure M with $D(M) = \mathbb{N}$ such that $M \models \mathcal{C}(\eta)$. By (C2) we have $\eta^M > 0$. Now let $m \in \mathbb{N}$, then we denote by $M[\eta \mapsto m]$ the $L \cup \{\eta\}$ structure with the same domain as M, that interprets all non-logical symbols except η as M does, and interprets η as m. Then we have $M[\eta \mapsto \eta^M - 1] \models \mathcal{C}(s(\eta))$ and by (C1) we now obtain $M[\eta \mapsto \eta^M - 1] \models \mathcal{C}(\eta)$. Hence, we obtain an infinite strictly descending sequence of natural numbers m such that $M[\eta \mapsto m] \models \mathcal{C}(\eta)$. This is impossible, hence $M \not\models \mathcal{C}(\eta)$.

Definition 6.1.3. *By $\mathsf{C}(\eta)$ we denote the $\mathcal{L}_{\mathrm{LA}} \cup \{\eta\}$ clause set*

$$cnf(B + \mathrm{B2}) \cup \{\{\eta \neq x + x\}, \{\eta \neq s(x + x)\}\}.$$

Example 6.1.4. *Intuitively, the clause set $\mathsf{C}(\eta)$ asserts the existence of an element η, which is neither even nor odd. We will now show that $\mathsf{C}(\eta)$ is a clause set cycle.*

We start by showing that $\mathsf{C}(\eta)$ *satisfies Condition* (C2). *Suppose that* $\mathsf{C}(0)$ *has a model* M, *then we have in particular* $M \models 0 \neq 0+0 = 0$. *This is a contradiction, and therefore* $\mathsf{C}(0) \models \bot$.

For Condition (C1), *let* M *be a model of* $\mathsf{C}(s(\eta))$. *Clearly, we have* $M \models \mathit{cnf}(B+\mathrm{B2})$, *hence we only have to show that* $M \models \eta \neq x + x$ *and* $M \models \eta \neq s(x + x)$. *Suppose that* $M \models \eta = d+d$ *for some* $d \in M$, *then we have* $M \models s(\eta) = s(d+d)$. *Since* $M \models \mathsf{C}(s(\eta))$, *we also have* $M \models s(\eta) \neq s(d+d)$, *a contradiction. Now suppose that* $M \models \eta = s(d+d)$ *for some* $d \in M$. *Since* $M \models \mathsf{C}(s(\eta))$, *we also have* $M \models s(\eta) = s(s(d+d)) = s(d)+s(d)$. *Thus* $M \models \mathsf{C}(\eta)$, *that is,* $\mathsf{C}(s(\eta)) \models \mathsf{C}(\eta)$.

Hence, $\mathsf{C}(\eta)$ *is a clause set cycle and therefore refutes itself.*

The induction argument contained in a refutation by a clause set cycle is peculiar in the sense that it does not take place in an explicit background theory. Instead of a background theory clause set cycles may contain clauses free of η that act as a background theory. In the example above the clause set cycle contains the clauses $\mathit{cnf}(B+\mathrm{B2})$, that correspond to the background theory. This phenomenon will be considered in more detail in Lemma 6.3.9.

Now let us briefly discuss how the notion of refutation by a clause set cycle may be integrated into a saturation-based theorem prover. This will be important to understand the scope of the unprovability results given in Section 6.4.

Definition 6.1.5. *The clause set cycle rule for saturation systems is given by*

$$\frac{C_1(\eta) \quad \dots \quad C_n(\eta)}{\square}\ \mathrm{CSC_S}$$

where $C_1, \dots, C_n$ *are clauses such that* $\{C_1, \dots, C_n\}$ *is a clause set cycle.*

Lemma 6.1.6. *Let* $\mathcal{S}$ *be a sound saturation system and let* $\mathcal{C}(\eta)$ *be a clause set. If* $\mathcal{C}$ *is refuted by* $\mathcal{S} + \mathrm{CSC_S}$, *then* $\mathcal{C}$ *is refuted by a* $\mathcal{L}(\mathcal{C}) \setminus \{\eta\}$ *clause set cycle.*

Proof. Let $\mathcal{C}_0, \dots, \mathcal{C}_n$ be a $\mathcal{S}+\mathrm{CSC_S}$ refutation. Without loss of generality we can assume that the $\mathrm{CSC_S}$ is applied only once and is used to derive the clause set $\mathcal{C}_n$. Hence, $n \geq 1$ and $\mathcal{C}_0, \dots, \mathcal{C}_{n-1}$ is a $\mathcal{S}$ deduction. By the soundness of $\mathcal{S}$ we have $\mathcal{L}(C_i) \subseteq \mathcal{L}(\mathcal{C}_0)$ for $i = 1, \dots, n$. Moreover, $\mathcal{C}_0 \models \mathcal{C}_{n-1}$ and there is a finite subset $\mathcal{D}$ of $\mathcal{C}_{n-1}$ such that $\mathcal{D}$ is a clause set cycle. Since $\mathcal{D} \subseteq \mathcal{C}_{n-1}$, we have $\mathcal{C}_{n-1} \models \mathcal{C}$. Hence, $\mathcal{D}$ is an $\mathcal{L}(\mathcal{C}_0)$ clause set cycle and $\mathcal{C}_0 \models \mathcal{D}$. □

Let $\mathcal{S}$ be a saturation system and consider the system $\mathcal{S} + \mathrm{CSC_S}$. In this system the rule $\mathrm{CSC_S}$ relies on the saturation system $\mathcal{S}$ to generate the clauses for which a clause set cycle may be detected. This situation is similar to the way that the rules considered in Chapters 4 and 5 rely on the underlying saturation system. Thus, we will qualify informally the rule $\mathrm{CSC_S}$ as analytic. Because of the analyticity of the clause set cycle detection rule, the converse of the lemma above does likely not hold.

By Lemma 6.1.6 we can approximate sound saturation systems extended by the rule $\mathrm{CSC_S}$ simply by the notion of refutation by a clause set cycle of Definition 6.1.2. Of

course, many practical systems extend the language of the prover for example by predicate symbols introduced for clause splitting. We do not consider such language extending mechanisms in this chapter and leave this as future work. Also observe that the clause set cycle rule given above is different from the induction rules for saturation systems that we have considered so far in that it is applied only once to terminate the refutation and does, in particular, not extend the working language of the prover by Skolem symbols. Furthermore, observe that the clause set cycle rule given above is also analytic in the sense that it detects only clause set cycles among derived clauses. As for the other methods considered so far we do not take into account this feature. Nevertheless we will obtain sufficiently simple unprovability results.

Since it is now clear how we intend that clause set cycles are used by saturation systems, we introduce some additional terminology in order to speak about formulas.

Definition 6.1.7. *Let φ be a sentence possibly containing the constant η, then we say that φ is proved by a clause set cycle (in symbols $\vdash_{\mathrm{CSC}} \varphi$) if $cnf(sk^{\exists}(\neg\varphi))$ is refuted by a $\mathcal{L}(sk^{\exists}(\neg\varphi)) \setminus \{\eta\}$ clause set cycle.*

The cycles detected by practical methods such as the n-clause calculus differ from clause set cycles in that they can be controlled by three parameters: An external offset, an internal offset, and the step size of the descent. Just as for the induction mechanisms considered in Chapter 4, these additional parameters allow the system to overcome some difficulties due to the inherent analyticity of rules like the clause set cycle rule given above. In the following we will show that when disregarding analyticity in the sense of Lemma 6.1.6, these parameters do not increase the overall strength of the system. In particular, a more detailed analysis taking into account the analyticity of a cycle detection rule may result in even more elementary unprovability results than those given in this thesis.

Definition 6.1.8. *Let $\mathcal{L}$ be a first-order language not containing η and $j, k \in \mathbb{N}$ with $j \geq 1$. A finite $\mathcal{L} \cup \{\eta\}$ clause set $\mathcal{C}(\eta)$ is called an $\mathcal{L}$ (j,k)-clause set cycle if*

$$\mathcal{C}(s^{j+k}(\eta)) \models \mathcal{C}(s^{k}(\eta)), \tag{C1'}$$

$$\mathcal{C}(\overline{m+k}) \models \bot, \text{ for } m = 0, \ldots, j-1. \tag{C2'}$$

We call the parameter j the internal offset and k the descent step size. Let $i \in \mathbb{N}$ and $\mathcal{D}(\eta)$ an $\mathcal{L} \cup \{\eta\}$ clause set, then $\mathcal{D}(\eta)$ is refuted by the (j,k)-clause set cycle $\mathcal{C}(\eta)$ with external offset i, if

$$\mathcal{D}(s^{i}(\eta)) \models \mathcal{C}(s^{k}(\eta)), \tag{C3'}$$

$$\mathcal{D}(\overline{m}) \models \bot, \text{ for } m = 0, \ldots, i-1. \tag{C3"}$$

Clearly, clause set cycles in the sense of Definition 6.1.2 are exactly the $(1,0)$-clause set cycles. A refutation by a clause set cycle in the sense of Definition 6.1.2 is a refutation by a $(1,0)$-clause set cycle with external offset 0.

We start by showing that (j,k)-clause set cycles with $j, k \in \mathbb{N}$ and $j \geq 1$ can be simulated by clause set cycles.

Lemma 6.1.9. *Let $\mathcal{L}$ be a first-order language not containing η, $j, k \in \mathbb{N}$ with $j \geq 1$, and $\mathcal{C}(\eta)$ an $\mathcal{L}$ (j,k)-clause set cycle. Then there exists a clause set cycle $\mathcal{C}'(\eta)$ such that $\mathcal{C}(s^k(\eta)) \models \mathcal{C}'(\eta)$.*

Proof. We start by eliminating the internal offset of the (j,k)-clause set cycle, by letting $\mathcal{C}'(\eta) \coloneqq \mathcal{C}(s^k(\eta))$. It is clear that $\mathcal{C}'$ is a $(j,0)$-clause set cycle. Moreover by the definition of $\mathcal{C}'$ we have $\mathcal{C}(s^k(\eta)) \models \mathcal{C}'(\eta)$. Let $\mathcal{C}''(\eta)$ be the clause set obtained by applying Lemma 2.4.5 to the set $\mathfrak{C} \coloneqq \{\mathcal{C}'(s^m(\eta)) \mid m = 0, \ldots, j-1\}$. We will now show that $\mathcal{C}''(\eta)$ is a clause set cycle. Suppose that $M \models \mathcal{C}''(0)$, then $M \models \mathcal{C}'(\overline{m})$ for some $m \in \{0, \ldots, j-1\}$, which is impossible and therefore $\mathcal{C}''(0) \models \bot$. Now suppose that $M \models \mathcal{C}''(s(\eta))$. Then we have $M \models \mathcal{C}'(s^{m+1}(\eta))$ for some $m \in \{0, \ldots, j-1\}$. If $m+1 \leq j-1$, then $\mathcal{C}(s^{m+1}(\eta)) \in \mathfrak{C}$ and therefore $M \models \mathcal{C}''(\eta)$. Otherwise we have $m+1 = j$ and therefore by C1' we obtain $M \models \mathcal{C}'(\eta)$ and therefore $M \models \mathcal{C}''(\eta)$. Since $\mathcal{C}'(\eta) \in \mathfrak{C}$, we have $\mathcal{C}'(\eta) \models \mathcal{C}''(\eta)$. □

Now we can show that a refutation by a (j,k)-clause set cycle with internal offset i, where $i, j, k \in \mathbb{N}$ with $j \geq 1$ can be reduced to a refutation by a clause set cycle.

Proposition 6.1.10. *Let $\mathcal{L}$ be a first-order language not containing η, $\mathcal{D}(\eta)$ an $\mathcal{L} \cup \{\eta\}$ clause set, and $i, j, k \in \mathbb{N}$ with $j \geq 1$. If $\mathcal{D}$ is refuted by an $\mathcal{L}$ (j,k)-clause set cycle with external offset i, then $\mathcal{D}(\eta)$ is refuted by a clause set cycle.*

Proof. Let $\mathcal{C}(\eta)$ be an $\mathcal{L}$ (j,k)-clause set cycle such that $\mathcal{D}(\eta)$ is refuted by $\mathcal{C}$ with external offset i. By Lemma 6.1.9 there exists a clause set cycle $\mathcal{C}'(\eta)$ such that $\mathcal{C}(s^k(\eta)) \models \mathcal{C}'(\eta)$. Hence $\mathcal{D}$ is refuted by a $(1,0)$-clause set cycle with external offset i. In the next step we will eliminate the external offset. Let $\mathfrak{C} \coloneqq \{\mathcal{D}(s^m(\eta)) \mid m = 0, \ldots, i-1\} \cup \{\mathcal{C}'(\eta)\}$ and apply Lemma 2.4.5 in order to obtain a clause set $\mathcal{C}''(\eta)$ corresponding to the disjunction of the clause sets in $\mathfrak{C}$. We will now show that $\mathcal{C}''(\eta)$ is a clause set cycle. Suppose that $M \models \mathcal{C}''(0)$, then either $M \models \mathcal{D}(s^m(\eta))$ for some $m \in \{0, \ldots, i-1\}$ or $M \models \mathcal{C}'(0)$. The first case is impossible because of Condition C3" and the second case is impossible because $\mathcal{C}'(\eta)$ is a clause set cycle and therefore $\mathcal{C}'(0) \models \bot$. Hence we have $\mathcal{C}''(0) \models \bot$. Now suppose that $M \models \mathcal{C}''(s(\eta))$. If $M \models \mathcal{C}'(s(\eta))$, then we have $M \models \mathcal{C}'(\eta)$ because $\mathcal{C}'$ is a clause set cycle and therefore $M \models \mathcal{C}''(\eta)$. If $M \models \mathcal{D}(s^{m+1}(\eta))$ for some $m \in \{0, \ldots, i-1\}$, we need to consider two cases. If $m+1 < i$, then we have $\mathcal{D}(s^{m+1}(\eta)) \in \mathfrak{C}$ and therefore $M \models \mathcal{C}''(\eta)$. Otherwise we have $m+1 = i$, and therefore we obtain $M \models \mathcal{C}'(\eta)$ by Condition C3'. Again we obtain $M \models \mathcal{C}''(\eta)$. Hence $\mathcal{C}''(\eta)$ is a clause set cycle. We complete the proof by observing that $\mathcal{D}(\eta) \models \mathcal{C}''(\eta)$, since $\mathcal{D}(\eta) \in \mathfrak{C}$. Hence, $\mathcal{D}(\eta)$ is refuted by the clause set cycle $\mathcal{C}''(\eta)$. □

Remark 6.1.11. *[HV20] uses a slightly different notation. A refutation by a clause set cycle in [HV20] corresponds to a refutation by a $(1,0)$-clause set cycle with external offset $i \in \mathbb{N}$. Hence, by Proposition 6.1.10 the notion of refutation by clause set cycle used in [HV20] is exactly as powerful as the more elegant notion of refutation by a clause set cycle used in this article.*

As already mentioned earlier, the notion of refutation by a clause set cycle is a useful intermediary abstraction of the induction mechanism of a family of AITP systems including in particular the n-clause calculus [KP13; Ker14]. Since our goal is to develop a uniform logical representation of methods for AITP, we thus use the notion of refutation by a clause set cycle as a starting point to provide logical abstractions of AITP systems such as the n-clause calculus. In particular, we want, for a fixed language $\mathcal{L}$ not containing η, to provide a logical theory T that simulates refutation by a clause set cycle in the following sense: Let $\mathcal{D}(\eta)$ be an $\mathcal{L} \cup \{\eta\}$ clause set that is refuted by an $\mathcal{L}$ clause set cycle, then $T + \mathcal{D}(\eta)$ is inconsistent.

In the next section we will consider some examples of what we can prove (refute) with clause set cycles. In particular this will show us that clause set cycles can prove sentences that are not provable by any the systems discussed in Chapters 4 and 5.

6.2 Induction over bounded and $\forall_1$ formulas

In this section we mention some examples that can be proved with clause set cycles but that can not be proved with some of the systems considered previously or with some related variants of induction. This will give us some idea of the power inherent to clause set cycles. In the first part of this section we give a concrete example based on the unprovability result given in Section 5.4. After that, in the second part we provide a variant of Parikh's theorem which provides us with a technique for obtaining examples that can be handled by clause set cycles but not by systems that infer only bounded formulas. This will in particular allow us to obtain Proposition 4.4 of [HV20] as a corollary.

We start by considering a variant of the clause set provided in Section 5.4 which is not refuted by the variant of Cruanes' calculus considered in Chapter 5 nor by the systems considered in Chapter 4.

Lemma 6.2.1. *The clause set* $cnf(\mathcal{A}_7 + (\forall y)\neg D_2(\eta, y))$ *is a clause set cycle.*

Proof. Let $\mathcal{C}(\eta) := cnf(\mathcal{A}_7 + (\forall y)\neg D_2(\eta, y))$. Suppose that $M \models \mathcal{C}(0)$, then by (A7.1) we have $M \models D_2(0,0)$ and $M \not\models D_2(0,0)$ because $M \not\models D_2(0,y)$. Now assume that $M \models \mathcal{C}(s(\eta))$ and let $d \in \mathcal{D}(M)$. In particular, we thus have $M \not\models D_2(s(\eta), s(s(d)))$. Hence by (A7.2) we obtain $M \models \neg D_2(\eta, d)$. Therefore, $M \models \mathcal{C}(\eta)$. □

The lemma above not only shows us that clause set cycles overcome the unprovability of Section 5.4 but also that the clause set which is not refuted by Cruanes' calculus is itself a clause set cycle and should thus be simple to detect for methods based on clause set cycles. However, we will show in Section 6.4 that clause set cycles are not in general stronger than Cruanes' calculus. We will even show that clause set cycles are not even as strong as induction over quantifier-free formulas.

The discrepancy between clause set cycles and Cruanes' system is essentially due to the incompatibility of induction for $\exists_1$ formulas and induction for $\forall_1$ formulas. However, as shown by Lemma 5.4.10 such examples do already not exist when the background

theory proves some basic facts about cut-off subtraction. In this case it makes sense to compare clause set cycles and weaker systems based on induction for quantifier-free formulas. By Shoenfield's theorem we readily obtain, for example, the following result.

Lemma 6.2.2. $B + \mathrm{Open}(\mathcal{L}_{\mathrm{LA}})\text{-IND} \nvdash \exists y(x = y + y \vee x = s(y + y))$

Proof. By Theorem 4.1.18 it suffices to show that $B' \nvdash \exists y(x = y + y \vee x = s(y + y))$. Consider the $\mathcal{L}_{\mathrm{LA}}$ structure M whose domain consists of the pairs of the form $(m, n) \in \mathbb{N} \times \mathbb{Z}$ such that $m = 0$ implies $n \in \mathbb{N}$ and that interprets the non-logical symbols as follows:

$$0^M = (0, 0),$$
$$s^M((m, n)) = (m, n + 1),$$
$$p^M((m, n)) = \begin{cases} (m, n \dot{-} 1) & \text{if } m = 0, \\ (m, n - 1) & \text{otherwise} \end{cases},$$
$$(m_1, n_1) +^M (m_2, n_2) = (m_1 + m_2, n_1 + n_2).$$

It is routine to verify that $M \models B'$. Consider the element $(1, 0)$, then clearly there is no element (m, n) of M such that $(1, 0) = (m, n) +^M (m, n) = (2m, 2n)$ or $(1, 0) = s^M((m, n) +^M (m, n)) = (2m, 2n + 1)$. □

By Example 6.1.4 we thus have an example where refutation by clause set cycles is stronger than induction for quantifier-free formulas.

By generalizing the argument used in [HV20] for the independence of the triangular number predicate by adapting Parikh's theorem ([Par71]) to our setting, we will now obtain a class of independence results for induction over bounded formulas (see Definition 6.2.5). For the sake of the completeness of the presentation we recall the basic model theoretic concepts and the proof of Parikh's theorem. The argument developed in the following provides quite a systematic way of showing that clause set cycles prove sentences that cannot be proved by systems that extend a Π_1 axiomatized base theory by true Π_1 formulas. In particular this allows us to show that clause set cycles are also stronger than induction over bounded formulas and anticipates unprovability results for the family of methods discussed in Chapter 8.

We work in a setting with a binary infix predicate symbol $\leq$, whose intended interpretation is the natural linear order of $\mathbb{N}$. The following definition describes the notion of substructure, that is, a structure that is included in some larger structure.

Definition 6.2.3. *Let $\mathcal{L}$ be a language and M, N $\mathcal{L}$ structures, then we call M a substructure of N, in symbols $M \subseteq N$ if $\mathcal{D}(M) \subseteq \mathcal{D}(N)$, $P^M = P^N \cap \mathcal{D}(M)^k$ for each predicate symbol $P/k \in \mathcal{L}$, and $f^M = f^N|_{\mathcal{D}(M)^k}$ for each function symbol f/k of $\mathcal{L}$.*

We will be particularly interested in substructures in which $\leq$ is downward-closed.

Definition 6.2.4. *The symbol $\leq$ is a binary predicate symbol. Let $\mathcal{L}$ be a language containing the symbol $\leq$ and M, N $\mathcal{L}$ structures. We say that M is an initial segment of*

N (or N is an end-extension of M, or M is a cut of N), in symbols $M \subseteq_e N$, if M is a substructure of N and for all $a \in \mathcal{D}(M)$ and $b \in \mathcal{D}(N)$, $N \models b \leq a$ implies $y \in \mathcal{D}(M)$.

These initial segments have the interesting property that the interpretation of bounded formulas does not change relative to end-extensions.

Definition 6.2.5. *Let φ be a formula, x a variable, and t a term not containing the variable x, then $(\exists x \leq t)\varphi$ is an abbreviation for $(\exists x)(x \leq t \wedge \varphi)$ and, similarly, $(\forall x \leq t)\varphi$ is an abbreviation for $(\forall x)(x \leq t \to \varphi)$. We call the quantifiers $(\exists x \leq t)$ and $(\forall x \leq t)$ bounded quantifiers. A formula is bounded if it only contains bounded quantifiers. Let $\mathcal{L}$ be a language containing the symbol $\leq$, then $\Sigma_0(\mathcal{L}) = \Pi_0(\mathcal{L}) = \Delta_0(\mathcal{L})$ is the set of bounded $\mathcal{L}$ formulas. Furthermore, $\Sigma_{n+1}(\mathcal{L})$ ($\Pi_{n+1}(\mathcal{L})$) is the set of formulas of the form $(\exists \vec{x})\varphi(\vec{x}, \vec{y})$ ($(\forall \vec{x})\varphi(\vec{x}, \vec{y})$) where φ is a $\Pi_n(\mathcal{L})$ ($\Sigma_n(\mathcal{L})$) formula and $\vec{x}$ is a possibly empty vector of variables.*

Note that the terms at the bounds of the quantifiers are $\mathcal{L}$ terms. For more precise statements one usually also distinguishes between the language of the formula and the language of the terms at bounds of quantifiers. However, for the purposes of this section such a distinction is not necessary.

Definition 6.2.6. *Let $\mathcal{L}$ be a language containing $\leq$, Γ a set of $\mathcal{L}$ formulas, and $M \subseteq N$ $\mathcal{L}$ structures, then M is a Γ-elementary substructure of N (in symbols $M \prec_\Gamma N$) if for all $\gamma(\vec{x}) \in \Gamma$ and $\vec{d} \in \mathcal{D}(M)^{|\vec{x}|}$, $M \models \gamma(\vec{d})$ if and only if $N \models \gamma(\vec{d})$.*

Lemma 6.2.7. *Let $\mathcal{L}$ be a language containing $\leq$, and M, N $\mathcal{L}$ structures such that $M \subseteq_e N$, then $M \prec_{\Delta_0(\mathcal{L})} N$.*

Proof. Let $\varphi(\vec{x})$ be a $\Delta_0(\mathcal{L})$ formula. We proceed by induction on the structure of φ and show that for all $\vec{d} \in M^{|\vec{x}|}$, $M \models \varphi(\vec{d})$ if and only if $N \models \varphi(\vec{d})$. If φ is an atom, then we clearly have $M \models \varphi(\vec{d})$ if and only if $N \models \varphi(\vec{d})$. If φ is of the form $\varphi_1 \wedge \varphi_2$, then by the induction hypothesis we have $M \models \varphi_i(\vec{d})$ if and only if $N \models \varphi_i(\vec{d})$ for $i = 1, 2$. Hence, we have $M \models \varphi_1(\vec{d}) \wedge \varphi_2(\vec{d})$ if and only if $N \models \varphi_1(\vec{d}) \wedge \varphi_2(\vec{d})$. We proceed similarly if φ is of the form $\varphi_1 \vee \varphi_2$ or of the form $\neg\varphi'$. If φ is of the form $(\exists y \leq t(\vec{z}))\varphi'(\vec{x}, y, \vec{z})$, then by the induction hypothesis we have $M \models \varphi'(\vec{d}, a, \vec{b})$, for all $\vec{d} \in \mathcal{D}(M)^{|\vec{x}|}$, $a \in \mathcal{D}(M)$, $b \in \mathcal{D}(M)^{|\vec{z}|}$. Let $\vec{d} \in \mathcal{D}(M)^{|\vec{x}|}$ and $b \in \mathcal{D}(M)$. Assume that $M \models (\exists y \leq t(\vec{b}))\varphi(\vec{d}, y, \vec{b})$, then there exits $a \in \mathcal{D}(M)$ such that $M \models a \leq t(\vec{b}) \wedge \varphi(\vec{d}, a, \vec{b})$. Since M is a substructure of N, we have $N \models a \leq t(\vec{b})$ and by the induction hypothesis $N \models \varphi(\vec{d}, a, \vec{b})$, thus, $N \models (\exists y \leq t(\vec{b}))\varphi(\vec{d}, y, \vec{b})$. Now assume that $N \models (\exists y \leq t(\vec{b}))\varphi(\vec{d}, y, \vec{b})$, then there exists $a \in \mathcal{D}(N)$ such that $N \models a \leq t(\vec{b}) \wedge \varphi(\vec{d}, a, \vec{b})$. Since $M \subseteq_e N$, we have $a \in \mathcal{D}(M)$ and therefore $M \models a \leq t(\vec{b})$. Furthermore, by the induction hypothesis we have $M \models \varphi(\vec{d}, a, \vec{b})$. We proceed analogously when φ is of the form $(\forall x \leq t(\vec{z}))\varphi'$. □

Lemma 6.2.8. *Let $\mathcal{L}$ be a language containing $\leq$, and let $M \subseteq_e N$ be $\mathcal{L}$ structures, $\varphi(\vec{x}) \in \Sigma_1(\mathcal{L})$, $\psi(\vec{x}) \in \Pi_1(\mathcal{L})$, and $\vec{a} \in \mathcal{D}(M)^{|\vec{x}|}$, then*

(i) $M \models \varphi(\vec{a})$ implies $N \models \varphi(\vec{a})$,

(ii) $N \models \psi(\vec{a})$ *implies* $M \models \psi(\vec{a})$.

Proof. An immediate consequence of Lemma 6.2.7. □

We are now ready to show the variant of Parikh's theorem that we will use for the separation of refutation by a clause set cycle from induction over bounded formulas. Let $\vec{t}, \vec{r}$ be finite sequences of terms of the same length, then we write $\vec{t} \leq \vec{r}$ as an abbreviation for $\bigwedge_{i=1}^{|\vec{t}|} t_i \leq r_i$. Similarly, if t' is a term, then $\vec{t} \leq t'$ is an abbreviation for $\bigwedge_{i=1}^{|\vec{t}|} t_i \leq t'$.

Theorem 6.2.9 (Variant of Parikh's theorem)**.** *Let $\mathcal{L}$ be a language containing the predicate symbol $\leq$, T be a set of $\Pi_1(\mathcal{L})$ sentences such that for all $\mathcal{L}$ terms $t_1, \ldots, t_n$ there exists a $\mathcal{L}$ term t' such that $T \vdash t_1, \ldots, t_n \leq t'$, $T \vdash x \leq y \wedge y \leq z \to x \leq z$, and for all function symbols f of $\mathcal{L}$ there is a $\mathcal{L}$ term $t_f(\vec{y})$ such that $T \vdash \vec{x} \leq \vec{y} \to f(\vec{x}) \leq t_f(\vec{y})$. Let $\psi(x, y)$ be a $\Delta_0(\mathcal{L})$ formula. If $T \vdash (\forall x)(\exists y)\psi(x, y)$, then there exists a $\mathcal{L}$ term $t(x)$ such that $T \vdash (\forall x)(\exists y \leq t(x))\psi(x, y)$.*

Proof. We proceed indirectly and assume that for all terms t, $T + (\exists x)(\forall y \leq t(x))\neg\psi(x, y)$ is consistent. Now let $t_1, t_2, \ldots,$ be an enumeration of the $\mathcal{L}$ terms and let $c/0$ be a function symbol not occurring in $\mathcal{L}$. We will show that the following theory T' is consistent

$$T + \{(\forall y \leq t_i(c))\neg\psi(c, y) \mid i \in \mathbb{N}\}.$$

Suppose that T' is inconsistent, then by the compactness theorem there is $m \in \mathbb{N}$ such that $T + \{(\forall y \leq t_i(c))\neg\psi(c, y) \mid 1 \leq i \leq m\}$ is inconsistent. By the properties of T there exists a term $t(x)$ such that $T \vdash t_1, \ldots, t_m \leq t$. Hence, $T + (\forall y \leq t(c))\neg\psi(c, y)$ is inconsistent as well. But this means that $T \vdash (\forall x)(\forall y \leq t(x))\psi(x, y)$, contradiction! Now let N be a model of T' and

$$\mathcal{M} \coloneqq \{a \in \mathcal{D}(N) \mid N \models a \leq t_i(c) \text{ for some } i \in \mathbb{N}\}.$$

We will now show that $\mathcal{M}$ induces an initial segment M of N. For a predicate symbol P/k of $\mathcal{L}$, we define $P^M = P^N \cap \mathcal{M}^k$. Now consider a function symbols f/k of $\mathcal{L}$ and let $f^M \coloneqq f^N|_{\mathcal{M}}$. We will show that the function f^M is closed on $\mathcal{M}$. Let $\vec{d} \in \mathcal{M}^k$, then there are terms $\vec{t} = (t_{i_1}(c), \ldots, t_{i_k}(c))$ such that $N \models d_i \leq t_{i_1}(c)$. By the monotonicity condition of f we thus have $N \models f(\vec{d}) \leq t_f(\vec{t})$. Since $t_f(\vec{t}) = t_i(c)$ for some $i \in \mathbb{N}$ we have $f(\vec{d}) \in \mathcal{M}$. Hence, M is a substructure of N. Now let $a \in \mathcal{D}(M)$ and $b \in \mathcal{D}(N)$ and assume that $N \models b \leq a$. Then there is an $i \in \mathbb{N}$ such that $N \models a \leq t_i(c)$, thus by transitivity of $\leq$ we also have $N \models b \leq t_i(c)$. Hence, $M \subseteq_e N$. By Lemma 6.2.8.(ii), M is a model of T, and moreover $M \models (\forall y)\neg\psi(c, y)$, thus $M \not\models (\forall x)(\exists y)\psi(x, y)$. Contradiction! □

Thanks to Theorem 6.2.9 we can now obtain many examples that allow us to separate refutation by a clause set cycle from other systems with induction. These examples are based on the idea that functions that grow fast enough are not provably total in T. Consider for example the following theory that extends the base theory of arithmetic with addition and a predicate symbol representing the graph function that assigns to n the n-th triangular number.

Definition 6.2.10. *Let $D_\triangleright$ be a binary predicate symbol. The theory $T_\triangleright$ extends the base theory of linear arithmetic B by the universal closure of the formulas*

$$D_\triangleright(0,0).$$
$$D_\triangleright(x,y) \to D_\triangleright(s(x), s(x)+y).$$

The theory $U_\triangleright$ extends the theory B by the universal closure of the formulas

$$D_\triangleright(0,y) \leftrightarrow y = 0,$$
$$D_\triangleright(s(x),y) \leftrightarrow (\exists v \leq y)[D_\triangleright(x,v) \wedge y = s(x)+v],$$
$$x \leq y \to (\exists z \leq y)(z + x = y),$$
$$(\exists z)(z + x = y) \to x \leq y.$$

Let $\mathcal{L}$ be a language containing $\leq$ and let M be a $\mathcal{L}$ structure, then we denote by $\Pi_n(M)$ the set of $\Pi_n(\mathcal{L})$ sentences that are true in M.

Proposition 6.2.11. $U_\triangleright + \Pi_1(\mathbb{N}_{\mathcal{L}(U_\triangleright)}) \nvdash (\forall x)(\exists y) x \triangleright y$

Proof. Let us abbreviate by T the theory $U_\triangleright + \Pi_1(\mathbb{N}_{\mathcal{L}(U_\triangleright)})$. Clearly, T proves that 0, s, $+$ are monotonic and $x \leq y \to p(x) \leq y$. Now proceed indirectly and assume that T proves $(\exists y) x \triangleright y$. Since T is $\Pi_1(\mathcal{L}(T))$ axiomatized, there exists by Theorem 6.2.9 a term $t(x)$ consisting of the symbols $0, s, +$ and p such that T proves $(\exists y \leq t(x)) x \triangleright y$. However, in the standard model t is bounded by a linear function and the triangular number grow strictly faster than a linear function. Contradiction! □

As a corollary we can also bound the growth rate of some provably total functions of $U_\triangleright + \Delta_0(\mathcal{L}(U_\triangleright))$-IND by terms of the language of $U_\triangleright$.

Corollary 6.2.12. $U_\triangleright + \Delta_0(\mathcal{L}(U_\triangleright))$-IND $\nvdash (\forall x)(\exists y) x \triangleright y$

Proof. It is straightforward to see that T has a $\Pi_1(\mathcal{L}(U_\triangleright))$ axiomatization, hence the claim follows immediately from Proposition 6.2.11. □

Definition 6.2.13. *We denote by $S_\triangleright(\eta)$ the clause set $cnf(T_\triangleright + (\forall y)\neg\eta \triangleright y)$.*

The clause set $S_\triangleright$ expresses that the triangle function is not total.

Lemma 6.2.14. *The clause set $S_\triangleright(\eta)$ is a $\mathcal{L}(T_\triangleright)$ clause set cycle.*

Proof. We have $S_\triangleright(0) \models D_\triangleright(0,0)$ and $S_\triangleright(0) \models \neg D_\triangleright(0,y)$. Hence $S_\triangleright(0) \models \bot$. Now assume $S_\triangleright(s(\eta))$. Since the clauses of $S_\triangleright$ that are free of η occur in $S_\triangleright(s(\eta))$, we only need to show that $S_\triangleright(s(\eta)) \models \neg D_\triangleright(\eta, y)$. Let y be arbitrary, then obtain $S_\triangleright(s(\eta)) \models \neg D_\triangleright(s(\eta), s(\eta)+y)$. Since $S(s(\eta)) \models D_\triangleright(x,y) \to D_\triangleright(s(x), s(x)+y)$, we have $\neg D_\triangleright(\eta, y)$. Therefore the clause set $S_\triangleright$ is a clause set cycle. □

Again since $S_\triangleright$ is a clause set cycle, this clause set is expected to be easy to detect by AITP systems that rely on clause set cycles, since there is no need to guess an intermediary clause set cycle. It is straightforward to see that $U_\triangleright + \Delta_0(\mathcal{L}(T_\triangleright))$-IND $\vdash T_\triangleright$. Hence, by Corollary 6.2.12, $S_\triangleright + \Delta_0(\mathcal{L}(T_\triangleright))$-IND is consistent.

Remark 6.2.15. *In Chapter 8 Parikh's theorem also immediately yields unprovability results for a family of theory exploration-based AITP systems.*

Thus clause set cycles are powerful enough to prove many interesting properties that cannot be proved with other systems. However, in Section 6.4 we will see that there are simple properties that can be proved even with quantifier-free induction but that cannot be proved with clause set cycles. In the next section we give a characterization of refutation by a clause set cycle in terms of a logical theory.

6.3 Logical characterization

In the previous sections we have introduced the notion of refutation by a clause set cycle and discussed how refutation by clause set cycles can be integrated into saturation-based provers. Furthermore, we have seen that clause set cycles are powerful enough to prove properties that cannot be proved with the systems described in Chapters 4 and 5. In this section we will give a characterization of refutation by a clause set cycle in terms of a logical theory. This characterization will be useful to provide unprovability results for clause set cycles and to consider possible extensions of the system that are perhaps not so easy to consider in terms of clause set cycles. The characterization that we develop in this section works over sets of clauses and does therefore not address the issue of the Skolem symbols introduced via clausification. However, the unprovability results of Section 6.4 are developed in a setting similar to the $\exists_2$ setting where clausification does not introduce additional Skolem symbols.

We start by converting clause set cycles into formulas. After that, we will discuss several remarkable properties of clause set cycles that will allow us to give a logical characterization of refutation by a clause set cycle.

Lemma 6.3.1. *Let $\mathcal{C}(\eta)$ be an $\mathcal{L}$ clause set cycle, then the formula $\neg cnf^{-1}(\mathcal{C})[\eta/x]$ is $\varnothing$-inductive. Let $\mathcal{D}(\eta)$ be an $\mathcal{L} \cup \{\eta\}$ clause set that is refuted by the clause set cycle $\mathcal{C}(\eta)$, then $\neg cnf^{-1}(\mathcal{C}) + \mathcal{D}(\eta)$ is inconsistent.*

Proof. Clearly, we have $M \models \neg cnf^{-1}(\mathcal{C})$ if and only if $M \not\models \mathcal{C}$. Hence, $\models \neg cnf^{-1}(\mathcal{C}(0))$ and $\neg cnf^{-1}(\mathcal{C}(\eta)) \models \neg cnf^{-1}(\mathcal{C}(s(\eta)))$. Therefore, by the completeness theorem and the deduction theorem for first-order logic we have

$$\vdash \neg cnf^{-1}(\mathcal{C}(0)),$$
$$\vdash \neg cnf^{-1}(\mathcal{C}(\eta)) \to \neg cnf^{-1}(\mathcal{C}(s(\eta))).$$

Thus, $\vdash \neg cnf^{-1}(\mathcal{C})[\eta/0]$ and $\vdash \neg cnf^{-1}(\mathcal{C})[\eta/x] \to \neg cnf^{-1}(\mathcal{C})[\eta/s(x)]$. The second part of the lemma is obvious. □

Let $\mathcal{C}$ be a clause set cycle, then by the lemma above the formula $\neg cnf^{-1}(\mathcal{C})[\eta/x]$ is the formula that corresponds to the induction argument contained in a refutation by a clause set cycle. In the following we will discern four important features of a refutation by a clause set cycle:

1. The formula $\neg cnf^{-1}(\mathcal{C})[\eta/x]$ is logically equivalent to an $\exists_1$ formula;
2. The only free variable of $\neg cnf^{-1}(\mathcal{C})[\eta/x]$ is the variable on which the argument by induction takes places. Hence the induction captured by clause set cycles is essentially parameter-free induction;
3. The formula $\neg cnf^{-1}(\mathcal{C})[\eta/x]$ is $\varnothing$-inductive. In a refutation by a clause set cycle, there is no explicit induction axiom. Instead, whenever a clause set $\mathcal{C}(\eta)$ is shown to be a clause set cycle, it can be used in a refutation. This is reminiscent of an unnested application of a Hilbert-style induction rule (see Definition 2.5.10) that allows us to deduce $\neg cnf^{-1}(\mathcal{C})[\eta/x]$ if $\neg cnf^{-1}(\mathcal{C})[\eta/x]$ is $\varnothing$-inductive;
4. Unlike an induction rule, a refutation by a clause set cycle makes use of the formula $\neg cnf^{-1}(\mathcal{C})$ instead of $(\forall x)(\neg cnf^{-1}(\mathcal{C})[\eta/x])$. In other words, refutations by a clause set cycle instantiate the conclusion of the induction rule only by η.

The first three points are unproblematic to express in the formalisms for induction that we have introduced so far. The restriction on the instances of conclusion of the induction rule are addressed by the following restricted induction rule.

Definition 6.3.2. *Let Γ be a set of formulas, then the rule Γ-IND$^R_\eta$ consists of the instances of the form*

$$\frac{\forall z \gamma(0, \vec{z}) \qquad \forall z(\gamma(x, \vec{z}) \to \gamma(s(x), \vec{z}))}{\forall \vec{z} \gamma(\eta, \vec{z})}, \text{ with } \gamma \in \Gamma.$$

We denote by Γ-IND$^{R-}_\eta$ the parameter-free counterpart of the rule Γ-IND$^R_\eta$.

By combining the above observations we obtain the following proposition, that allows us to simulate clause set cycles in a logical theory.

Proposition 6.3.3. *Let $\mathcal{D}(\eta)$ be an $\mathcal{L} \cup \{\eta\}$ clause set. If $\mathcal{D}(\eta)$ is refuted by an $\mathcal{L}$ clause set cycle, then $[\varnothing, \exists_1(\mathcal{L})\text{-IND}^{R-}_\eta] \cup \mathcal{D}(\eta)$ is inconsistent.*

Proof. Since $\mathcal{D}$ is refuted by a clause set cycle, there exists an $\mathcal{L}$ clause set cycle $\mathcal{C}(\eta)$ such that

$$\mathcal{D}(\eta) \models \mathcal{C}(\eta). \tag{*}$$

Let $\varphi(x) \coloneqq cnf^{-1}(\mathcal{D})[\eta/x]$ then $\varphi(\eta)$ is clearly logically equivalent to $\mathcal{D}(\eta)$. By the soundness of first-order logic it thus suffices to show that

$$[\varnothing, \exists_1(\mathcal{L})\text{-IND}^{R-}] \vdash \neg\varphi(\eta).$$

Let $\psi(x)$ be an $\exists_1$ formula that is logically equivalent to $\neg cnf^{-1}(\mathcal{C})[\eta/x]$. Then, by applying the completeness theorem and the deduction theorem to (*), we obtain

$$\vdash \varphi(\eta) \to \neg\psi(\eta). \tag{†}$$

By Lemma 6.3.1 we know that $\psi(x)$ is $\varnothing$-inductive, and therefore we have $[\varnothing, \exists_1(\mathcal{L})\text{-IND}^{R-}_{\eta}] \vdash \psi(\eta)$. Hence, by considering the contrapositive of (†) we clearly obtain $[\varnothing, \exists_1(\mathcal{L})\text{-IND}^{R-}_{\eta}] \vdash \neg\varphi(\eta)$. □

We will now show that we even have the converse and thus obtain a characterization of refutation by a clause set cycle by a logical theory. We start by observing that finitely many inductive formulas can be fused into a single inductive formula.

Lemma 6.3.4. *Let T be a theory and let $\varphi_1(x, \vec{z})$, …, $\varphi_n(x, \vec{z})$ be formulas. If φ_i is T-inductive in x for $i = 1, \ldots, n$, then $\psi \coloneqq \bigwedge_{i=1,\ldots n} \varphi_i$ is T-inductive in x.*

Proof. We start by showing that $T \vdash \psi$. Let $j \in \{1, \ldots, n\}$, then since φ_j is T-inductive in x, we have $T \vdash \varphi_j(0, \vec{z})$ and we are done. Now let us show that $T \vdash \psi(x, \vec{z}) \to \psi(s(x), \vec{z})$. Work in T, assume $\bigwedge_{i=1}^{n} \varphi_i(x, \vec{z})$, and let $j \in \{1, \ldots, n\}$. Since φ_j is T-inductive in x, we have $\varphi_j(x, \vec{z}) \to \varphi_j(s(x), \vec{z})$. Hence we obtain $\varphi_j(x, \vec{z})$ and therefore ψ is T-inductive in x. □

This simple result is particularly interesting because fusing inductive formulas neither introduces more induction parameters and when fusing $\exists_k$ induction formulas, the fused induction formula is also logically equivalent to an $\exists_k$ formula. Similar techniques exist for fusing a finite number of induction axioms into a single induction axiom [HW18; Gen54]. However, these either introduce a new induction parameter or increase the quantifier complexity of the resulting induction formula.

Proposition 6.3.5. *Let $\mathcal{D}(\eta)$ be an $\mathcal{L} \cup \{\eta\}$ clause set. If $[\varnothing, \exists_1(\mathcal{L})\text{-IND}^{R-}_{\eta}] + \mathcal{D}(\eta)$ is inconsistent, then $\mathcal{D}(\eta)$ is refuted by an $\mathcal{L}$ clause set cycle.*

Proof. Let $\varphi(x) \coloneqq cnf^{-1}(\mathcal{D})[\eta/x]$, then by the completeness theorem and the deduction theorem we obtain $[\varnothing, \exists_1(\mathcal{L})\text{-IND}^{R-}_{\eta}] \vdash \neg\varphi(\eta)$. By the compactness theorem there exist $\exists_1$ $\mathcal{L}$ formulas $\psi_1(x)$, …, $\psi_k(x)$ such that ψ_i is $\varnothing$-inductive for $i = 1, \ldots, k$ and

$$\psi_1(\eta) + \cdots + \psi_k(\eta) \vdash \neg\varphi(\eta).$$

By Lemma 6.3.4, the formula $\Psi(x) \coloneqq \bigwedge_{i=1}^{k} \psi_i$ is $\varnothing$-inductive. Moreover, we have $\Psi(\eta) \vdash \neg\varphi(\eta)$. Clearly, Ψ is logically equivalent to an $\exists_1$ formula, hence there exists a $\forall_1$ formula Θ that is logically equivalent to $\neg\Psi$. Since $\vdash \Psi(0)$ and $\vdash \Psi(x) \to \Psi(s(x))$, we have $\Theta(0) \models \bot$ and $\Theta(s(x)) \models \Theta(x)$. Therefore, $\mathcal{C} \coloneqq cnf(\Theta(\eta))$ is a clause set cycle. Finally, since $\Psi(\eta) \vdash \neg\varphi(\eta)$, we obtain $\varphi(\eta) \models \neg\Psi(\eta)$, that is, $\mathcal{D}(\eta) \models \mathcal{C}(\eta)$. In other words, $\mathcal{D}$ is refuted by the clause set cycle $\mathcal{C}$. □

We thus obtain a characterization of refutation by a clause set cycle in terms of induction rules.

Theorem 6.3.6. *Let $\mathcal{L}$ be language not containing η and $\mathcal{D}$ an $\mathcal{L} \cup \{\eta\}$ clause set, then $\mathcal{D}$ is refuted by an $\mathcal{L}$ clause set cycle if and only if $[\varnothing, \exists_1(\mathcal{L})\text{-IND}^{R-}_{\eta}] + \mathcal{D}$ is inconsistent.*

Proof. An immediate consequence of Propositions 6.3.3 and 6.3.5. □

Remark 6.3.7. *In a refutation by a clause set cycle the constant η plays essentially two roles: On the one hand, it can be thought of as a Skolem symbol and, on the other hand, it plays the role of an induction variable. The characterization of Theorem 6.3.6 clarifies this situation by allowing us to distinguish between induction variables and the Skolem symbol η.*

The characterization of Theorem 6.3.6 allows us to straightforwardly consider various extensions of refutation by a clause set cycle that are perhaps not so straightforward to consider in terms of clause set cycles. For example, it becomes now easy to consider a system that lifts the restriction on the instances of the induction rule, or to consider systems that allow nested applications of the induction rule. As a corollary we obtain Theorem 2.10 of [HV20].

Corollary 6.3.8 ([HV20, Theorem 2.10])**.** *Let $\mathcal{L}$ be a first-order language not containing η and $\mathcal{D}$ an $\mathcal{L}\cup\{\eta\}$ clause set. If $\mathcal{D}$ is refuted by an $\mathcal{L}$ clause set cycle, then $\exists_1(\mathcal{L})$-IND$+\mathcal{D}$ is consistent.*

Proof. Obvious, since $\exists_1(\mathcal{L})\text{-IND} \vdash [\varnothing, \exists_1(\mathcal{L})\text{-IND}^{R-}_{\eta}]$. □

In the following we slightly reformulate Theorem 6.3.6 so as to facilitate the handling of formulas and theories. In Section 6.1 we have informally observed that arguments by clause set cycles do not take place in some explicit background theory and that instead a clause set cycle contains the clauses corresponding to the background theory. In the following we will make this informal observation more precise by working with induction rules.

Lemma 6.3.9. *Let $\mathcal{L}$ be a first-order language, T a $\forall_1$ $\mathcal{L}$ theory, U an $\mathcal{L}$ theory, then*

$$T + [U, \exists_1(\mathcal{L})\text{-IND}^{R-}_{\eta}] \equiv [T + U, \exists_1(\mathcal{L})\text{-IND}^{R-}_{\eta}].$$

Furthermore, $T + [U, \exists_1(\mathcal{L})\text{-IND}^{R-}] \equiv [T + U, \exists_1(\mathcal{L})\text{-IND}^{R-}]$.

Proof. The direction $[T + U, \exists_1(\mathcal{L})\text{-IND}^{R-}_{\eta}] \vdash T + [U, \exists_1(\mathcal{L})\text{-IND}^{R-}_{\eta}]$ is immediate. For the other direction let $\gamma(x)$ be an $\exists_1$ $\mathcal{L}$ formula and assume that $T + U \vdash \gamma(0)$ and $T + U \vdash \gamma(x) \to \gamma(s(x))$. By the compactness theorem and the deduction theorem there exist $\tau, \tau_1, \ldots, \tau_n \in T$ such that $\tau = \bigwedge_{i=1}^{n} \tau_i$ and $U \vdash \tau \to \gamma(0)$ and $U \vdash \tau \to \gamma(x) \to \gamma(s(x))$. By straightforward propositional equivalences we obtain

$$U \vdash (\tau \to \gamma(x)) \to (\tau \to \gamma(s(x)))\,.$$

Clearly, τ is logically equivalent to a $\forall_1$ sentence, hence $\tau \to \gamma(x)$ is logically equivalent to an $\exists_1$ formula $\gamma'(x)$. Hence, $[U, \exists_1(\mathcal{L})\text{-IND}^{R-}_{\eta}] \vdash \gamma'(\eta)$ and therefore $[U, \exists_1(\mathcal{L})\text{-IND}^{R-}] \vdash \tau \to \gamma(\eta)$. Thus, $T + [U, \exists_1(\mathcal{L})\text{-IND}^{R-}_{\eta}] \vdash \gamma(\eta)$. We show $T + [U, \exists_1(\mathcal{L})\text{-IND}^{R-}] \equiv [T+U, \exists_1(\mathcal{L})\text{-IND}^{R-}]$ analogously, with the exception that in the last part of the argument we have to shift the universal quantifier in $\forall x(\tau \to \gamma(x))$ inwards. □

Lemma 6.3.9 allows us to move $\forall_1$ axioms in and out of the induction rule. This allows us to consider the η-free clauses of a clause set cycle as the background theory. As an immediate consequence Lemma 6.3.9 and Theorem 6.3.6 we now obtain a general pattern to reduce unrefutability problems for clause set cycles to independence problems. Note that we focus on a setting that is reminiscent of the $\exists_2$ setting that we have considered in Chapters 3 to 5.

Corollary 6.3.10. *Let $\mathcal{L}$ be a first-order language not containing η, T a $\forall_1$ $\mathcal{L}$ theory, and $\varphi(x, \vec{y})$ a quantifier-free $\mathcal{L}$ formula, then $[T, \exists_1(\mathcal{L})\text{-IND}^{R-}_{\eta}] \vdash (\exists\vec{y})\varphi(\eta, \vec{y})$ if and only if $cnf(T + (\forall\vec{y})\neg\varphi(\eta, \vec{y}))$ is refuted by an $\mathcal{L}$ clause set cycle.*

Proof. Clearly, $T + [\varnothing, \exists_1(\mathcal{L})\text{-IND}^{R-}_{\eta}] \vdash \exists\vec{y}\varphi(\eta, \vec{y})$ if and only if $[\varnothing, \exists_1(\mathcal{L})\text{-IND}^{R-}_{\eta}] + cnf(T + \forall\vec{y}\neg\varphi(\eta, \vec{y}))$ is inconsistent. By Theorem 6.3.6, $[\varnothing, \exists_1(\mathcal{L})\text{-IND}^{R-}_{\eta}] + cnf(T + \forall\vec{y}\neg\varphi(\eta, \vec{y}))$ is inconsistent if and only if $cnf(T + \forall\vec{y}\neg\varphi(\eta, \vec{y}))$ is refuted by an $\mathcal{L}$ clause set cycle. Since T is a $\forall_1$ $\mathcal{L}$ theory, the theory $T + [\varnothing, \exists_1(\mathcal{L})\text{-IND}^{R-}_{\eta}]$ is Lemma 6.3.9 equivalent to $[T, \exists_1(\mathcal{L})\text{-IND}^{R-}_{\eta}]$ □

We conclude this section with the observation that parameter-free induction rule for $\exists_1$ formulas contains the parameter-free induction schema for quantifier-free formulas.

Lemma 6.3.11. *Let $\mathcal{L}$ be a language, then $[\varnothing, \exists_1(\mathcal{L})\text{-IND}^{R-}] \vdash \text{Open}(\mathcal{L})\text{-IND}^{-}$.*

Proof. Let $\varphi(x)$ be a quantifier-free formula, then it suffices to observe that the formula $\varphi(0) \wedge (\forall x)(\varphi(x) \to \varphi(s(x))) \to (\forall x)\varphi(x)$ is $\varnothing$-inductive and is logically equivalent to an $\exists_1$ formula. □

In the following section we will make use of the characterization of Theorem 6.3.6 to construct clause sets that are refutable by open induction but which are not refutable by clause set cycles. In particular the unrefutability results that we provide exploit different logical features of clause set cycles.

6.4 Unprovability by clause set cycles

In the previous sections we have introduced the notion of refutation by a clause set cycle and we have seen that refutation by a clause set cycle naturally overcomes some of the unprovability phenomena that affect the methods considered in Chapters 4 and 5. Moreover, we have provided a characterization of refutation by a clause set cycle in terms of a logical theory. We have shown this characterization by discerning four main logical features of refutation by a clause set cycle: the quantifier-complexity of the argument by infinite descent, the absence of induction parameters, the similarity with induction rules, and the restriction on instances of derived formulas. In this section we will make use of this characterization in order to provide several clause sets that are not refutable by clause set cycles but that are refutable in related systems. These unprovability suggest various extensions of systems based on the detection of clause set cycles. The unrefutability results in this section will exploit different logical features of clause set cycles.

In Section 6.4.1 we will provide an unprovability result that shows that restricting the instances of the conclusion of the induction rule can be very drastic. In Section 6.4.2 we will provide an unprovability result that essentially relies on the absence of induction parameters from clause set cycles. Furthermore, in Section 6.4.3 we will consider a hierarchy of systems that allow for increasingly deeply nested applications of the induction rule and we show that allowing for more deeply nested applications of the induction rule may result in increasingly strong systems. Finally, in Section 6.4.4 we provide an unprovability result that separates systems based on the induction rule and systems based on the induction axiom.

6.4.1 Instance restriction

In Section 6.3 we have observed that a refutation by a clause set cycle only permits a single instance of a clause set cycle to appear in a refutation. In this section we will formulate an unprovability result for clause set cycles that exploits this restriction. In particular, we will base this unprovability result on a stronger independence result that shows how drastic the instance restriction is.

Definition 6.4.1. *The theory $\mathcal{P}$ is axiomatized by the universal closure of the following formulas*

$$\begin{gathered} 0 \neq s(x), \\ s(x) = s(y) \to x = y, \\ P(0), \\ P(x) \to P(s(x)). \end{gathered}$$

Definition 6.4.2. *Let $\varphi(x, \vec{z})$ be a formula, then $I^{\eta}_{x}\varphi$ denotes the formula*

$$\varphi(0, \vec{z}) \wedge \forall x(\varphi(x, \vec{z}) \to \varphi(s(x), \vec{z})) \to \varphi(\eta, \vec{z}).$$

Let Γ be a set of formulas, then the theory Γ-IND$_\eta$ is axiomatized by the universal closure of the formulas $I^{\eta}_{x}\gamma$ with $\gamma \in \Gamma$.

We have the following independence.

Proposition 6.4.3. *Let $f/1$ be a function symbol with $f \neq 0$ and $f \neq s$, then $\mathcal{P} + \mathcal{F}(\{0, s, P, f\})$-IND$_\eta \not\vdash P(f(\eta))$.*

Proof. Let M be the $\{0, s, P, f\}$ structure with domain consisting of pairs $(m, n) \in \{0, 1\} \times \mathbb{Z}$ such that if $m = 0$, then $n \in \mathbb{N}$. Let M interpret the non-logical symbols as follows

$$\begin{gathered} 0^M = \eta^M = (0, 0), \\ s^M((m, n)) = (m, n + 1), \\ f^M((m, n)) = (1, n), \\ P^M = \{(0, n) \mid n \in \mathbb{N}\}. \end{gathered}$$

It is clear that M is a $\{0, s, P, f\}$ structure and moreover it is straightforward to verify that M is a model of $\mathcal{P}$. Now let us show that $M \models \mathcal{F}(\{0, s, P, f\})\text{-IND}_\eta$. Let $\psi(x, \vec{z})$ be a $\{0, s, P, f\}$ formula, $\vec{c}$ a vector of elements of M. Assume that $M \models \psi(0, \vec{c})$ and $M \models \psi(x, \vec{c}) \to \psi(s(x), \vec{c})$. Since $\eta^M = 0^M$, we already have $M \models \psi(\eta, \vec{c})$ and therefore $M \models I^\eta_x \psi(x, \vec{z})$. Finally, observe that $f^M(\eta^M) = (1, 0) \notin P^M$, hence $\mathcal{P}+\mathcal{F}(\{0, s, P, f\})\text{-IND}_\eta \nvdash P(f(\eta))$. □

The above independence result is remarkable in the sense that it imposes no restriction whatsoever on the induction formulas, only the conclusion of the induction axioms is restricted. Hence the result shows that this restriction is extremely strong. As a corollary we obtain the following unrefutability result for clause set cycles.

Corollary 6.4.4. *The $\{0, s, P, f, \eta\}$ clause set $cnf(\mathcal{P} + \neg P(f(\eta)))$ is not refuted by a $\{0, s, P, f\}$ clause set cycle.*

Proof. Suppose that $cnf(\mathcal{P} + \neg P(f(\eta)))$ is refuted by a $\{0, s, P, f\}$ clause set cycle. Then, by Corollary 6.3.10 we have $[\mathcal{P}, \exists_1(\{0, s, P, f\})\text{-IND}^{R-}_\eta] \vdash P(f(\eta))$. However, since $\mathcal{P}+\mathcal{F}(\{0, s, P, f\})\text{-IND}_\eta \vdash [\mathcal{P}, \exists_1(\{0, s, P, f\})\text{-IND}^{R-}_\eta]$, this contradicts Proposition 6.4.3. □

Lemma 6.4.5. $[\mathcal{P}, \text{Open}(\{0, s, P, f\})\text{-IND}^{R-}] \vdash P(f(\eta))$.

Proof. The formula $P(x)$ is inductive in $\mathcal{P}$. □

Proposition 6.4.3, Corollary 6.4.4, and Lemma 6.4.5 together show that the η-restriction as encountered in the n-clause calculus is drastic and can result in pathological unrefutability phenomena. On the one hand, without the η-restriction a very simple argument by induction suffices to prove $P(f(\eta))$ and on the other hand in presence of the η-restriction even induction for all $\{0, s, P, f\}$ formulas does not allow us to prove the formula $P(f(\eta))$. However, because of this the unrefutability result of Corollary 6.4.4 does not tell us anything about the other restrictions of the induction principle contained in refutations by a clause set cycle.

Hence, it would be interesting to have a similar result for linear arithmetic. In particular we conjecture the following.

Conjecture 6.4.6. $[B, \exists_1(\mathcal{L}_{\text{LA}})\text{-IND}^{R-}_\eta] \nvdash 0 + (\eta + \eta) = (\eta + \eta)$.

6.4.2 Absence of induction parameters

In the following we will consider another unprovability result for clause set cycles that does not make use of the instance restriction, but instead exploits the lack of induction parameters in clause set cycles. This time we work in the setting of linear arithmetic described in Section 2.6. The unprovability result developed in this section is based on the following weak cancellation property of the addition of natural numbers.

Definition 6.4.7. *Let $k, n, m \in \mathbb{N}$ with $0 < n < m$, then we define*

$$n \cdot x + \overline{(m-n)k} = m \cdot x \to x = \overline{k}. \qquad (\text{E}_{k,n,m})$$

The formula $\mathrm{E}_{k,n,m}$ is a generalization of

$$x + 0 = x + x \to x = 0. \tag{$\mathrm{E}_{0,1,2}$}$$

Most of the upcoming Chapter 7 is devoted to the proof of the following independence result.

Theorem 6.4.8. *Let $n, m, k \in \mathbb{N}$ with $0 < n < m$, then*

$$B + \mathrm{B2} + \mathrm{B3} + \exists_1(\mathcal{L}_{\mathrm{LA}})\text{-IND}^- \nvdash \mathrm{E}_{k,n,m}.$$

By making use of the above independence result and the characterization of refutation by a clause set cycle in Corollary 6.3.10, we straightforwardly obtain an unrefutability result.

Definition 6.4.9. *Let $k, n, m \in \mathbb{N}$ with $0 < n < m$, then we define the clause set $\mathcal{E}_{k,n,m}(\eta)$ by $cnf(B + \mathrm{B2} + \mathrm{B3} + \neg\mathrm{E}_{k,n,m}(\eta))$.*

Corollary 6.4.10. *Let $k, n, m \in \mathbb{N}$ with $0 < n < m$, then the clause set $\mathcal{E}_{k,n,m}(\eta)$ is not refuted by an $\mathcal{L}_{\mathrm{LA}}$ clause set cycle.*

Proof. Assume that $cnf(B + \mathrm{B2} + \mathrm{B3} + \neg\mathrm{E}_{k,n,m}(\eta))$ is refuted by a clause set cycle. By Corollary 6.3.10 we have $[B + \mathrm{B2} + \mathrm{B3}, \exists_1(\mathcal{L}_{\mathrm{LA}})\text{-IND}^{R-}_{\eta}] \vdash \mathrm{E}_{k,n,m}(\eta)$. Clearly, this contradicts Theorem 6.4.8. □

Furthermore, the clause sets $\mathcal{E}_{k,n,m}(\eta)$ with $k, n, m \in \mathbb{N}$ and $0 < n < m$ are refuted by open induction.

Proposition 6.4.11. $\mathrm{Open}(\mathcal{L}_{\mathrm{LA}})\text{-IND} \cup \mathcal{E}_{k,n,m}(\eta)$ *is inconsistent.*

Proof. Clearly, it suffices to show that $B + \mathrm{Open}(\mathcal{L}_{\mathrm{LA}})\text{-IND} \vdash \mathrm{E}_{k,n,m}(x)$. Work in $B + \mathrm{Open}(\mathcal{L}_{\mathrm{LA}})\text{-IND}$ and assume $n \cdot x + \overline{(m-n)k} = m \cdot x$. Then by (B2), (B3), and (B4) we obtain $\overline{(m-n)k} = (m-n) \cdot x$. Now we use (B1) to proceed by case analysis on x. If $x = \overline{k'}$ with $k' < k$, then we have $\overline{(m-n)(k-k')} = 0$. Since $m - n > 0$ and $k - k' > 0$ this contradicts Lemma 2.6.9.(i). If $x = \overline{k}$, then we are done. If $x = s^{k+1}(p^{k+1}(x))$, then $0 = (m-n) + p^{k+1}(x)$, which contradicts Lemma 2.6.9.(i). □

Hence, Corollary 6.4.10 together with Proposition 6.4.11 give a positive answer to Conjecture 4.7 of [HV20]. We conclude this section with some remarks on this result and possible improvements.

The formula $E_{0,1,2}(x)$ is particularly interesting, because it can be proven by a comparatively straightforward induction.

Lemma 6.4.12. $[B, \mathrm{Open}(\mathcal{L}_{\mathrm{LA}})\text{-IND}^R] \vdash E_{0,1,2}$.

Proof. Clearly it suffices to show that the formula $\varphi(x, y) \coloneqq x + 0 = y + x \to y = 0$ is B-inductive in x. It is obvious that $B \vdash \varphi(0, y)$. Now work in B and assume $\varphi(x, y)$ and $s(x) + 0 = y + s(x)$. By (A4) and (A5) we obtain $s(x + 0) = s(x) = s(x) + 0 = y + s(x) = s(y + x)$. By Lemma 2.2.5 we obtain $x + 0 = y + x$, hence by the assumption we obtain $y = 0$. □

This demonstrates that clause set cycles are a very weak induction mechanism in the sense that they are unable to deal even with simple generalizations and therefore fail to refute relatively simple clause sets. The unprovability results in Corollaries 6.4.4 and 6.4.10 were constructed so that only one Skolem constant η appears in the language of the considered clause sets. Consider now the clause set $\mathcal{C}$ given by

$$cnf(B) \cup \{\{\eta + 0 = \nu + \eta\}\} \cup \{\{\nu \neq 0\}\},$$

where ν is a Skolem constant distinct from η. It is straightforward to check that $\mathcal{C}(\eta)$ is an $\mathcal{L}_{\mathrm{LA}} \cup \{\nu\}$ clause set cycle. Hence, if clause set cycles are detected on the languages obtained by Skolemization of the given property and its background theory, then clause set cycles allow us to prove the property $x + 0 = y + x \to y = 0$ from B but fail to prove the weaker property $x + 0 = x + x \to x = 0$ from B. Thus, clause set cycles are sensitive to the syntactic material present in a given set of clauses. Moreover, provability by a clause set cycle is thus not deductively closed in the sense in the sense that $\vdash_{\mathrm{CSC}}$ is not closed under modus ponens. Let $\varphi(x, y)$ be the formula

$$x + 0 = y + x \to y = 0,$$

then by the above we have $\vdash_{\mathrm{CSC}} (\forall y)\varphi(\eta, y)$. Moreover, since $\vdash (\forall y)\varphi(\eta, y) \to \varphi(\eta, \eta)$ we have $\vdash_{\mathrm{CSC}} (\forall y)\varphi(\eta, y) \to \varphi(\eta, \eta)$, but $\nvdash_{\mathrm{CSC}} \varphi(\eta, \eta)$ by Corollary 6.4.10. Hence, provability by clause set cycles is not deductively closed.

The independence result of Theorem 6.4.8 also shows that the unrefutability result of Corollary 6.4.10 does neither rely on the η-restriction nor on the induction rule like nature of clause set cycles. Thus the unprovability may not be overcome by extensions of clause set cycles that only address these points.

Furthermore, we believe that an independence similar to the one in Theorem 6.4.8 also holds for the atomic formula $x + (x + x) = (x + x) + x$, which is a well-known challenging formula for inductive theorem provers [BIS92; Bee06; Haj+20].

Conjecture 6.4.13. $B + \exists_1(\mathcal{L}_{\mathrm{LA}})\text{-IND}^- \nvdash x + (x + x) = (x + x) + x$.

Below we provide an abstraction of the unprovability result given in Corollary 6.4.10. This abstraction shows the structure of the problem more clearly and allows us to focus on the features that are responsible for the unprovability. In particular the abstract formulation allows us to relate unprovability results for clause set cycles with the unprovability result for the single-clause induction rule considered in Section 4.3.

Definition 6.4.14. *Let $A/2$ be a predicate symbol, then the theory $\mathcal{A}_5$ is axiomatized by the universal closure of the following formulas*

$$A(0, y), \tag{A5.1}$$
$$A(x, y) \to A(s(x), y). \tag{A5.2}$$

Corollary 6.4.15. $\mathcal{A}_5 + \exists_1(\mathcal{L}(\mathcal{A}_5))\text{-IND}^- \nvdash A(x, x)$.

Proof. Let $\varphi(x, y)$ be the formula given by $x + 0 = y + x \to y = 0$. Let M be an $\mathcal{L}_{\mathrm{LA}}$ structure that witnesses the independence Theorem 6.4.8. Now let N be the structure that expands M by the following interpretation of the predicate A:

$$A^N = \{(a, b) \mid a, b \in \mathcal{D}(M), M \models \varphi(a, b)\}.$$

Since $\varphi(x, y)$ is $(B + \mathrm{B2} + \mathrm{B3})$-inductive in x, we have $N \models A(0, y)$ and $N \models A(x, y) \to A(s(x), y)$. Thus, $N \models \mathcal{A}_5$, Furthermore, every quantifier-free $\mathcal{L}(\mathcal{A}_5)$ is equivalent in N to a quantifier-free $\mathcal{L}_{\mathrm{LA}}$ formula. Hence, $N \models \exists_1(\mathcal{L}(\mathcal{A}_5))\text{-IND}^-$, but of course $N \not\models A(x, x)$. □

In Section 4.3 we will again briefly consider this abstract formulation when discussing some possible improvements of our results by taking into account the analyticity of realistic AITP systems.

6.4.3 Nesting depth of the induction rule

In this section we briefly consider the role of the depth of the nesting of applications of the induction rule. We will show that a formalism that extends clause set cycles to achieve at most a fixed finite number of nestings of the corresponding induction rule will have an unprovable clause set, that becomes provable when the nesting depth is increased by one. Moreover, the result remains valid in extensions of clause set cycles that allow for induction parameters. However, the unprovability results in this section are more abstract than in the previous sections in the sense that we work with a much stronger background theory. We expect that providing more elementary unprovability results is not difficult but this is left as future work.

In the remainder of the section we will show the following result.

Theorem 6.4.16. *Let $k \in \mathbb{N}$, then there is a language $\mathcal{L}$ not containing η and an $\mathcal{L} \cup \{\eta\}$ clause set $\mathcal{C}(\eta)$ such that $\mathcal{C}$ is consistent with $[\varnothing, \exists_1(\mathcal{L})\text{-IND}^{R-}]_k$ but inconsistent with $[\varnothing, \exists_1(\mathcal{L})\text{-IND}^{R-}]_{k+1}$.*

The language of Peano arithmetic $\mathcal{L}_{\mathrm{PA}}$ consists of the function symbols $0/0$, $s/1$, the infix function symbols $+/2$, $*/2$, and the infix predicate symbol $\leq /2$.

Definition 6.4.17. *We let $\Sigma_n = \Sigma_n(\mathcal{L}_{\mathrm{PA}})$, $\Pi_n = \Pi_n(\mathcal{L}_{\mathrm{PA}})$, and $\Delta_0 = \Delta_0(\mathcal{L}_{\mathrm{PA}})$.*

Definition 6.4.18. *Let T be a theory, $\varphi(x_1, \ldots, x_k, y)$ a formula, and $f : \mathbb{N}^k \to \mathbb{N}$ a function. We say that f has definition $\varphi(x_1, \ldots, x_k, y)$ in T if for all $n_1, \ldots, n_k, m \in \mathbb{N}$, $T \vdash \varphi(\overline{n_1}, \ldots, \overline{n_k}, \overline{m})$ if and only if $f(n_1, \ldots, n_k) = m$. Furthermore, we say that a formula $\psi(\vec{x}, y)$ defines a total function in T if $T \vdash (\forall \vec{x})(\exists! y)\varphi(\vec{x}, y)$. Finally, the function f is provably total recursive in T if f has Σ_1 definition $\psi(\vec{x}, y)$ in T that defines a total function in T.*

We will prove the theorem above by providing a sequence of theories $T_0, T_1, \ldots$ with $\mathcal{L}(T_i) \supseteq \mathcal{L}_{\mathrm{PA}}$, $T_{i+1} = [T_i, \exists_1(\mathcal{L}(T_0))\text{-IND}^{R-}]$ such that the provably total recursive functions of T_i are exactly those of the level $3 + i$ of the Grzegorczyk hierarchy, for $i \in \mathbb{N}$,

and over T_0 the Σ_0 formulas are exactly the $\exists_1(\mathcal{L}(T_0))$ formulas. Since, the Grzegorczyk hierarchy is a strict hierarchy (see for example [Ros84]), we obtain for each level $i \in \mathbb{N}$ a quantifier-free $\mathcal{L}(T_0)$ formula $\varphi(x, y)$, such that $(\exists y)\varphi(x, y)$ is provable in T_{i+1} but not in T_i. For a definition of the Grzegorczyk hiearchy we refer the reader to [Ros84].

Definition 6.4.19. *Let $n \in \mathbb{N}$, then we denote by $\mathcal{E}_n$ the n-th level of the Grzegorczyk hiearchy.*

The background theory is Robinson arithmetic.

Definition 6.4.20 (Robinson arithmetic)**.** *The theory Q is axiomatized by the universal closure of the following axioms*

$$s(x) \neq 0, \tag{Q1}$$
$$s(x) = s(y) \to x = y, \tag{Q2}$$
$$x \neq 0 \to (\exists y)(x = s(y)), \tag{Q3}$$
$$x + 0 = x, \tag{Q4}$$
$$x + s(y) = s(x + y), \tag{Q5}$$
$$x * 0 = 0, \tag{Q6}$$
$$x * s(y) = (x * y) + x, \tag{Q7}$$
$$x \leq y \leftrightarrow (\exists z)(z + x = y). \tag{Q8}$$

Definition 6.4.21. *Let $n \in \mathbb{N}$, then the theory $Q + \Sigma_n$-IND is called $I\Sigma_n$. The theory $I\Sigma_0$ is also called $I\Delta_0$.*

There is a Δ_0 definition of the exponential function such that the theory $I\Delta_0$ proves the inductive properties of the definition of the exponential function, but by Parikh's theorem (see Theorem 6.2.9) it is clear that $I\Delta_0$ does not prove the totality of such a definition of the exponential function.

Lemma 6.4.22. *There is a Δ_0 formula $\mathrm{Exp}(x, y, z)$ such that $I\Delta_0$ proves*

$$\mathrm{Exp}(x, 0, z) \leftrightarrow z = \overline{1}, \tag{E1}$$
$$\mathrm{Exp}(x, s(y), z) \leftrightarrow (\exists v)(\mathrm{Exp}(x, y, v) \wedge z = v * x). \tag{E2}$$

In particular $I\Delta_0$ proves $\mathrm{Exp}(x, y, z_1) \wedge \mathrm{Exp}(x, y, z_2) \to z_1 = z_2$.

Proof. See [HP93, Section V.3] □

In the following we will mainly work a theory that extends $I\Delta_0$ extended by the totality of the exponential function.

Definition 6.4.23. *By $I\Delta_0 + \mathrm{EXP}$ we denote the theory that extends $I\Delta_0$ by the axiom $(\forall x)(\forall y)(\exists z)\mathrm{Exp}(x, y, z)$.*

The theory $I\Delta_0 + \mathrm{EXP}$ is also called elementary arithmetic and has various equivalent formulations, see [Bek05, Section 1.1]. In the following we will develop a particular formulation with a $\forall_1$ axiomatization and in which the $\exists_1$ formulas of the extended language are exactly the Σ_1 formulas.

Lemma 6.4.24. *$I\Delta_0$ has a Π_1 axiomatization.*

Proof. Drop axiom Q3, replace axiom Q8 by the universal closure of the formulas $x \leq y \to (\exists z \leq y)z + x = y$ and $z + x = y \to x \leq y$, and replace the induction axioms $I_x\varphi$, where $\varphi(x, \vec{z})$ is Δ_0 by

$$(\varphi(0, \vec{z}) \wedge (\forall y < x)(\varphi(y, \vec{z}) \to \varphi(s(y), \vec{z}))) \to \varphi(x, \vec{z}).$$

It is routine to check that the resulting theory is equivalent to $I\Delta_0$. □

Now we will show that $I\Delta_0$ has Δ_0 definitions of Skolem functions of all Δ_0 formulas. Later on we will introduce the corresponding Skolem functions in order to get rid of bounded quantifiers.

Lemma 6.4.25. *$I\Delta_0$ proves the least number principle for Δ_0 formulas.*

Proof. See [HP93, Theorem 1.22]. □

By using the least number principle for Δ_0 formulas, we can provide in $I\Delta_0$ definitions for Skolem functions of Δ_0 predicates of the form $(\exists z \leq y)\varphi(\vec{x}, y, z)$. The Skolem function is defined by selecting for fixed $\vec{x}$ and y the least element z such that $z \leq y$ and $\varphi(\vec{x}, y, z)$ or by letting $z = 0$ if no such element exists.

Definition 6.4.26. *Let $\varphi(\vec{x}, y, z)$ be a Δ_0 formula, then the formula $D_{(\exists z \leq y)\varphi}(\vec{x}, y, z)$ is given by*

$$\big(z \leq y \wedge \varphi(\vec{x}, y, z) \wedge (\forall z' < z)\neg\varphi(\vec{x}, y, z')\big) \vee \big((\forall z' \leq y)\neg\varphi(\vec{x}, y, z') \wedge z = 0\big)\,.$$

Lemma 6.4.27. *Let $\varphi(\vec{x}, y, z)$ be a Δ_0 formula, then $I\Delta_0$ proves*

(i) $(\exists z \leq y)\varphi(\vec{x}, y, z) \to (\exists z \leq y)(D_{(\exists z \leq y)\varphi}(\vec{x}, y, z) \wedge \varphi(\vec{x}, y, z))$

(ii) $(\forall z \leq y)\neg\varphi(\vec{x}, y, z) \to D_{(\exists z \leq y)\varphi}(\vec{x}, y, 0)$

(iii) $(\exists! z)D_{(\exists z \leq y)\varphi}(\vec{x}, y, z)$,

(iv) $D_{(\exists z \leq y)\varphi}(\vec{x}, y, z) \to z \leq y$.

Proof. The formula 6.4.27.(i) follows easily from Lemma 6.4.25. The formulas 6.4.27.(ii)–6.4.27.(iv) are straightforward. □

We will now define the way in which we Skolemize Δ_0 formulas. We Skolemize Δ_0 functions from inside out in order to avoid having to introduce Skolem symbols for formulas that contain Skolem symbols.

Definition 6.4.28. *Let $\varphi(\vec{x}, y, z)$ be a Δ_0 formula, then $F_{(\exists z \leq y)\varphi}$ is a function symbol of arity $|\vec{x}| + 1$.*

Definition 6.4.29. *The formula translations $(\cdot)^{\exists}$ and $(\cdot)^{\forall}$ are defined mutually recursively by*

$$(\theta)^Q = \theta, \text{ if } \theta \text{ is quantifier-free},$$
$$(\varphi_1 \wedge \varphi_2)^Q = \varphi_1^Q \wedge \varphi_2^Q,$$
$$(\varphi_1 \vee \varphi_2)^Q = \varphi_1^Q \vee \varphi_2^Q,$$
$$(\neg\varphi)^Q = \neg\varphi^{\overline{Q}},$$
$$((\overline{Q}y \leq x)\varphi)^Q = (\overline{Q}y \leq x)\varphi^Q$$
$$((\exists y \leq x)\varphi(x, y, \vec{z}))^{\exists} = \left(y \leq x \wedge \varphi^{\exists}\right)[y/F_{(\exists y \leq x)\varphi}(x, \vec{z})], \tag{$*_1$}$$
$$((\forall y \leq x)\varphi(x, y, \vec{z}))^{\forall} = \left(y \leq x \to \varphi^{\forall}\right)[y/F_{(\exists y \leq x)\neg\varphi}(x, \vec{z})], \tag{$*_2$}$$

where $Q \in \{\forall, \exists\}$, $\overline{\forall} = \exists$, $\overline{\exists} = \forall$, and in Eqs. $(*_1)$ and $(*_2)$ the variables $\vec{z}$ all appear freely in the formula φ.

We can now obtain a suitable formulation of $I\Delta_0 + \mathrm{EXP}$.

Lemma 6.4.30. *There exists a $\forall_1$ axiomatized conservative extension T of $I\Delta_0 + \mathrm{EXP}$ such that every $\exists_1(\mathcal{L}(T))$ formula is equivalent over T to a Σ_1 formula and every Σ_1 formula is equivalent over T to an $\exists_1(\mathcal{L}(T))$ formula.*

Proof. We consider a Π_1 formulation U of $I\Delta_0$. For each axiom $(\forall\vec{x})\varphi$ of U where φ is Δ_0, T contains the axiom $(\forall\vec{x})\varphi^{\exists}$. Furthermore, T contains the axiom $\mathrm{Exp}^{\exists}[z/e(x, y)]$. Finally, for each Δ_0 formula $\varphi(\vec{x}, y, z)$, T contains the axiom $(D_{(\exists z \leq y)\varphi})^{\exists}[z/F_{(\exists z \leq y)\varphi}(\vec{x}, y)]$. Now obtain a $\forall_1$ axiomatization by moving the remaining quantifiers outwards. By a model-theoretic argument it is straightforward to see that the resulting theory is conservative over $I\Delta_0 + \mathrm{EXP}$.

It is straightforward to check that every Δ_0 formula φ is equivalent in T to a quantifier-free $\mathcal{L}(T)$ formula. Let ψ be a Σ_1 formula, then $\psi = (\exists\vec{x})\varphi$ where φ is Δ_0. Hence, ψ is equivalent over T to the formula $(\exists\vec{x})\varphi'$ where φ' is a quantifier-free formula that is equivalent over T to φ. Now let ψ be an $\exists_1(\mathcal{L}(T))$ formula, then by [Hod97, pp. 51–52] there exists an equivalent unnested $\exists_1(\mathcal{L}(T))$ formula of the form $(\exists\vec{x})\varphi$ where φ is quantifier-free. Now we simply replace atoms of the form $f(\vec{u}) = y$ where f is either a Skolem symbol of a Δ_0 formula or e by the corresponding defining Δ_0 formula. Hence, the resulting formula is a Σ_0 formula. □

In the following we fix one such extension of $I\Delta_0 + \mathrm{EXP}$ and call it EA.

Definition 6.4.31. *Let $k \in \mathbb{N}$, then EA_k denotes the theory $[\mathrm{EA}, \Pi_2\text{-IND}^R]_k$.*

Theorem 6.4.32 ([Sie91])**.** *The provably total recursive functions of the theory EA_k are precisely those of the class $\mathcal{E}_{3+k}$ of the Grzegorczyk hierarchy.*

Proof. See also the proof Corollary 7.5 of [Bek97a]. □

We can reformulate the theories EA_k as follows.

Lemma 6.4.33. *Let $k \in \mathbb{N}$, then* $\mathrm{EA}_k \equiv [\mathrm{EA}, \exists_1(\mathcal{L}(\mathrm{EA}))\text{-IND}^{R-}]_k$.

Proof. We proceed by induction on k and show

$$\mathrm{EA}_k \equiv [\mathrm{EA}, \exists_1(\mathcal{L}(\mathrm{EA}))\text{-IND}^{R-}]_k.$$

If $k = 0$, then the claim follows trivially. Now assume the claim for k, then EA_k is Π_2 axiomatized, hence by [Bek97a, Corollary 7.4]

$$\mathrm{EA}_{k+1} \equiv [\mathrm{EA}_k, \Pi_2\text{-IND}^R] \equiv [\mathrm{EA}_k, \Sigma_1\text{-IND}^R].$$

Furthermore, by [Bek05, Lemma 4.6] we have

$$[\mathrm{EA}_k, \Sigma_1\text{-IND}^R] \equiv [\mathrm{EA}_k, \Sigma_1\text{-IND}^{R-}].$$

Since over EA the Σ_1 formulas are exactly the $\exists_1(\mathcal{L}(\mathrm{EA}))$ formulas, we obtain

$$[\mathrm{EA}_k, \Sigma_1\text{-IND}^{R-}] \equiv [\mathrm{EA}_k, \exists_1(\mathcal{L}(\mathrm{EA}))\text{-IND}^{R-}].$$

By the induction hypothesis we readily obtain

$$[\mathrm{EA}, \Pi_2\text{-IND}^R]_{k+1} \equiv [\mathrm{EA}, \exists_1(\mathcal{L}(\mathrm{EA}))\text{-IND}^{R-}]_{k+1}.$$ □

Since $\mathcal{E}_k \subsetneq \mathcal{E}_{k+1}$ for all $k \in \mathbb{N}$, we can now provide a proof of Theorem 6.4.16.

Proof of Theorem 6.4.16. Let $k \in \mathbb{N}$, then there exists a function $f : \mathbb{N} \to \mathbb{N}$ such that $f \in \mathcal{E}_{k+4} \setminus \mathcal{E}_{k+3}$. Hence, there exists a Σ_1 formula $\varphi(x, y)$ such that $f(n) = m$ if and only if $\mathbb{N} \models \varphi(\overline{n}, \overline{m})$ and

$$[\mathrm{EA}, \exists_1(\mathcal{L}(\mathrm{EA}))\text{-IND}^{R-}]_{k+1} \vdash (\exists y)\varphi(x, y),$$
$$[\mathrm{EA}, \exists_1(\mathcal{L}(\mathrm{EA}))\text{-IND}^{R-}]_k \nvdash (\exists y)\varphi(x, y).$$

Thus, by the construction of EA, there exists a quantifier-free $\mathcal{L}(\mathrm{EA})$ formula $\varphi'(x, y, \vec{z})$ such that $\mathrm{EA} \vdash \varphi \leftrightarrow (\exists \vec{z})\varphi'$. Since EA is $\forall_1$ axiomatized we furthermore have $\mathrm{EA} + [\varnothing, \exists_1(\mathcal{L}(\mathrm{EA}))\text{-IND}^{R-}] \equiv [\mathrm{EA}, \exists_1(\mathcal{L}(\mathrm{EA}))\text{-IND}^{R-}]$. Hence, the clause set

$$cnf(\mathrm{EA} + (\forall y)(\forall \vec{z})\neg\varphi'(\eta, y, \vec{z}))$$

has the desired properties. This completes the proof. □

This result tells us that a mechanism that extends refutation by a clause set cycle so as to allow at most k-fold nested $\exists_1$ parameter-free induction rule is strictly weaker than a mechanism that allows $(k+1)$-fold nested applications of the $\exists_1$ parameter-free induction rule. This naturally gives rise to the question whether we can separate a system that

provides arbitrary nestings of the parameter-free $\exists_1$ induction rule from a system that provides the parameter-free $\exists_1$ induction schema. The following lemma shows that we need a different approach to resolve this question.

Lemma 6.4.34 ([Par72]). *$I\Sigma_1$ is Π_2 conservative over* $\mathrm{EA} + \Sigma_1\text{-IND}^R$.

Hence the theory $\mathrm{EA}+\exists_1(\mathcal{L}(\mathrm{EA}))\text{-IND}$ is also Π_2 conservative over $\mathrm{EA}+\mathcal{L}(\mathrm{EA})\text{-IND}^{R-}$. Thus the technique used above does not provide us with a clause set that separates both systems. In Section 6.4.4 we provide a simple clause set that is consistent with nested applications of the parameter-free $\exists_1$ induction rule but not with the corresponding induction schema.

The results in this section are less elementary than the results of Sections 6.4.1 and 6.4.2 in the sense that we work over the comparatively strong EA. Moreover, we work with an infinite axiomatization of EA over an infinite language. The last point can be improved by selecting the Skolem symbol more carefully. We leave finding more elementary examples that separate the systems formed by successive applications of the parameter-free induction rule for $\exists_1$ formulas as future work. Nevertheless, the results are interesting because of the connection with the Grzegorczyk hiearchy, which tells us that successive applications of the induction rule allow us to deal with more primitive recursive functions.

6.4.4 Induction rule

In this section we will show that even an extension of clause set cycles by a mechanism that allows for an unrestricted nesting depth of applications of the parameter-free $\exists_1$ induction rule will have some simple unprovability examples that are overcome when working with the corresponding induction schema. The result that we provide in this section is slightly different from the results of the previous sections in the sense that the result relies on the consistency of a theory rather than an independence result. However, the result is interesting because it shows us that clause set cycles interact less with the context in which the argument takes place than the induction schema. This is an advantage of the systems considered in Chapters 3 to 5.

Definition 6.4.35. *Let $P/2$ be a predicate symbol. Let $\mathcal{L}_P$ be the language $\mathcal{L}_0 \cup \{P/2\}$. The theory $\mathcal{A}_8$ extends the theory $\mathcal{A}_0$ by the following axioms*

$$P(0,\eta), \tag{C1}$$
$$(\forall x)(\forall y)(P(x,y) \to P(s(x),y)), \tag{C2}$$
$$(\forall y)\neg P(\eta,y). \tag{C3}$$

In this section we will show the following result.

Theorem 6.4.36. *$\mathcal{A}_8 + \bigcup_{i<\omega}[\varnothing, \exists_1(\mathcal{L}_P)\text{-IND}^{R-}]_i$ is consistent.*

As an immediate consequence we obtain an unprovability result for clause set cycles.

Corollary 6.4.37. *The clause set $cnf(\mathcal{A}_8)$ is not refuted by a $\mathcal{L}_P$ clause set cycle.*

Proof. Proceed indirectly and assume that $cnf(\mathcal{A}_8)$ is refuted by an $\mathcal{L}_P$ clause set cycle. Hence, by Theorem 6.3.6 the theory $\mathcal{A}_8 + [\varnothing, \exists_1(\mathcal{L}_P)\text{-IND}^{R-}]$ is inconsistent. However, this contradicts Theorem 6.4.36. □

On the other hand we have the following.

Lemma 6.4.38. *$\mathcal{A}_8 + \exists_1(\mathcal{L}_P)\text{-IND}^-$ is inconsistent.*

Proof. Work in $\mathcal{A}_8 + \exists_1(\mathcal{L}_P)\text{-IND}^-$. We proceed by induction on the formula $(\exists y)P(x,y)$. For the base case we have $P(0,\eta)$ from $\mathcal{A}_8$ and therefore $(\exists y)P(0,y)$. For the induction step assume that there is some y such that $P(x,y)$. By (C2) we thus obtain $P(s(x),y)$, that is, $(\exists y)P(s(x),y)$. Hence, we obtain $(\forall x)(\exists y)P(x,y)$, which contradicts (C3). □

Roughly speaking, the additional power of the induction schema comes the interaction of the induction axioms with the background theory. Even though all the axioms of $\mathcal{A}_8$ are $\forall_1$ axioms, the induction rule $\exists_1(\mathcal{L}_P)\text{-IND}^{R-}$ does not have access to these axioms because they contain the constant η. This corresponds to how clause set cycles handle the special Skolem constant η. Interestingly, as we show in the following, even the variant of the induction rule rigid single-clause induction rule (see Definition 4.1.1) given below is sufficient to refute the theory $\mathcal{A}_8$. Let the rule $\text{RSCIND}'_{\text{S}}$ consist of the instances of the form

$$\frac{\overline{L}(a) \vee C}{\{\neg L(0) \vee L(c) \vee L(x), \neg L(0) \vee \neg L(s(c)) \vee L(x)\},}\ \text{RSCIND}'_{\text{S}}$$

where $sk^{\forall}((\forall x)(L(x) \to L(s(x)))) = L(c) \to L(s(c))$, $L(x)$ is a literal free of the constant a. Consider now the clause set $\mathcal{C}_0 = \{P(0,y), \neg P(x,y) \vee P(s(x),y), \neg P(\eta,y)\}$, then the induction rule $\text{RSCIND}'_{\text{S}}$ can be applied to the clause $P(0,y)$ to derive in particular the clause

$$P(0,\eta) \vee \neg P(c_1,\eta) \vee \neg P(x,\eta),$$

where $sk^{\forall}((\forall x)(\neg P(x,\eta) \to \neg P(s(x),\eta))) = \neg P(c_1,\eta) \to \neg P(s(c_1),\eta)$. Now we may apply the induction rule $\text{RSCIND}'_{\text{S}}$ to the literal $\neg P(c,\eta)$ on the constant c in the clause above in order to obtain the clauses

$$\neg P(0,\eta) \vee P(c_2,\eta) \vee P(x,\eta),$$
$$\neg P(0,\eta) \vee \neg P(s(c_2),\eta) \vee P(x,\eta).$$

where $sk^{\forall}((\forall x)(P(x,\eta) \to P(s(x),\eta)) = P(c_2,\eta) \to P(s(c_2),\eta))$. Now resolve these clauses with the initial clauses in $\mathcal{C}_0$ in order to derive the empty clause.

In the remainder of this section we will provide a proof of Theorem 6.4.36. The proof essentially consist of the the simplification of $\exists_1(\mathcal{L}(\mathcal{A}_8))$ formulas and the construction of a suitable first-order structure.

Let us start by introducing the notion of co-clauses and components. Moreover, components will also be of use for the arguments in Section 7.2.

Definition 6.4.39 (Co-clauses and Components)**.** *A co-clause is a conjunction of literals. A component $\chi(x)$ is a formula of the form $\exists\vec{y}C_\chi(x,\vec{y})$, where C_χ is a co-clause.*

Lemma 6.4.40. *Let $\varphi(x)$ be a $\mathcal{L}_P$ formula of the form $\bigvee_{i=1}^n \chi_i(x)$ where $\chi_1,\ldots,\chi_n$ are components. If φ is inductive in the theory*

$$[\mathcal{A}_0 + \mathrm{ACY}, \exists_1(\mathcal{L}_P)\text{-IND}^{R-}]_k,$$

then either $\vdash \varphi(x)$ or φ has a disjunct χ_i with $i \in \{1,\ldots,n\}$ that neither contains a positive P literal nor $\bot$.

Proof. Let M be the $\mathcal{L}_P$ structure with domain $\mathbb{N}$ that interprets the symbols of $\mathcal{L}_P$ as follows: $0^M = 0$, $s^M(n) = n+1$, and $P^M = \varnothing$. By the choice of the domain it is obvious that $M \models [\mathcal{A}_0 + \mathrm{ACY}, \exists_1(\mathcal{L}_P)\text{-IND}^{R-}]_k$. Assume that $\varphi(x)$ is inductive in $[\mathcal{A}_0 + \mathrm{ACY}, \exists_1(\mathcal{L}_P)\text{-IND}^{R-}]_k$. Then, in particular $M \models \varphi(0)$. If $n = 0$, then $\vdash \varphi(x)$ and we are done. Otherwise, if there is a positive P literal or $\bot$ in every χ_i with $i = 1,\ldots,n$, then by the definition of M we have $M \models (\forall x)(\forall y)\neg P(x,y)$, hence $M \not\models \varphi(0)$. □

Lemma 6.4.41. *Let $C(x,\vec{y})$ be a $\mathcal{L}(\mathcal{A}_8)$ co-clause, then there exists N and a co-clause $C'(x,\vec{y})$ free of positive equational literals involving the variables $\vec{y}$ such that over $\mathcal{A}_0 +$ ACY, $(\exists\vec{y})C(s^N(x),\vec{y})$ is equivalent to $(\exists\vec{y})C'(x,\vec{y})$.*

Proof. Proceed by lexicographic induction on the pair (p,q) where p is the number of variables of $\vec{y}$ that occur in C and q is the number of positive literals of C. If C contains a literal of the form $s^{k_1}(y_i) = s^{k_2}(y_j)$ with $k_1 \leq k_2$, then there are two cases. If $i = j$, then we decide the literal over $\mathcal{A}_0 + \mathrm{ACY}$. If $k_1 = k_2$, then we obtain C' by removing the literal from C and applying the induction hypothesis. Otherwise, we let $N = 0$ and $C' = \bot$ and we are done. If $i \neq j$, then over $\mathcal{A}_0$ the above literal is equivalent to $y_i = s^{k_2-k_1}(y_j)$ and C is equivalent to $y_i = s^{k_2-k_1}(y_j) \wedge C$. Hence, over $\mathcal{A}_8$, $(\exists\vec{y})C$ is equivalent to $(\exists\vec{y})C[y_i/s^{k_2-k_1}]$. The co-clause $C[y_i/s^{k_2-k_1}]$ does not contain y_i, hence we can apply the induction hypothesis. If C contains a literal of the form $s^{k_1}(x) = s^{k_2}(y_i)$, then let N be large enough such that $N + k_1 \geq k_2$. Then over $\mathcal{A}_0 +$ ACY, $(\exists\vec{y})C[x/s^N(x)]$ is equivalent to

$$(\exists\vec{y})C[x/s^N(x), y_i/s^{N+k_1-k_2}(x)].$$

Hence, we can apply the induction hypothesis. □

Lemma 6.4.42. *Let $\varphi(x)$ be an $\exists_1$ $\mathcal{L}(\mathcal{A}_8)$ formula, then there is $N \in \mathbb{N}$ and a disjunction of components $\psi(x)$ such that $\psi(x)$ is equivalent over $\mathcal{A}_0 +$ ACY to $\varphi(s^N(x))$ and ψ does neither contain ground literals, positive equational literals, nor negative equational literals in one variable.*

Proof. By Lemma 6.4.41 there is $N \in \mathbb{N}$ and $\psi(x) = \bigvee_{i=1}^n (\exists\vec{y})C_i(x,\vec{y})$ where $C_1,\ldots,C_n$ are co-clauses and $\vec{y} = (y_1,\ldots,y_m)$ such that the positive literals of ψ are free of the variables $y_1,\ldots,y_m$. Hence, only the variable x may occur in the positive literals of ψ. Since $\mathcal{A}_0 +$ ACY is complete for $\mathcal{L}_0$ literals in one variable, we simply remove from ψ co-clauses containing a literal l in one variable such that $\mathcal{A}_0 +$ ACY proves the dual and we remove those literals in one variable that are proved by $\mathcal{A}_0 +$ ACY. □

We are now ready to carry out the model-theoretic construction.

Definition 6.4.43. *Let M_0 be the $\mathcal{L}_P \cup \{\eta\}$ structure whose domain consists of pairs of the form $(i,n) \in \{0,1\} \times \mathbb{Z}$ such that $i = 0$ implies $n \in \mathbb{N}$ and that interprets the non-logical symbols as follows: $0^{M_0} = (0,0)$, $s^{M_0}(i,n) = (i,n+1)$, $P^{M_0} = \{((0,n),(1,0)) \mid n \in \mathbb{N}\}$, and $\eta^{M_0} = (1,0)$.*

Lemma 6.4.44. $M_0 \models \mathcal{A}_8$.

Proof. Straightforward. □

Lemma 6.4.45. *Let $C(\vec{x},\vec{y}) = \bigwedge_{i=1}^{n} l_i(\vec{x},\vec{y})$ be a co-clause with $\vec{x} = (x_1,\ldots,x_m)$ such that C is free of positive literals and does not contain negative equational literals in one variable. Let, furthermore, $\vec{a}$ be a finite sequence of elements of M_0 such that for all $i \in \{1,\ldots,n\}$, $Var(l_i) \subseteq \{x_1,\ldots,x_m\}$ implies $M_0 \models l_i(\vec{a})$, then for $B \in \mathbb{Z}$ there are integers $b_{|y|} \leq \cdots \leq b_2 \leq b_1 \leq B$ such that*

$$M_0 \models C(\vec{a},(1,b_1),\ldots,(1,b_{|y|})).$$

Proof. Let $\vec{y} = (y_1,\ldots,y_l)$, then we proceed by induction on l. If $l = 0$, then we are done. For the induction step we start by considering the equational literals whose free variables are among the variables $\{x_1,\ldots,x_m,y_1\}$ and contain an occurrence of y_1. Up to symmetry of equality these are of the forms $t(\vec{x}) \neq s^k(y_1)$. Let $M_0,[\vec{x}/\vec{a}] \models t = (i,p)$, then it is not difficult to see that there is a C such that for $c \leq C$ we have $M_0 \models (i,p) \neq s^{k'}((1,c))$. Take for example $C = p - k' - 1$. Hence, there is an integer $C \leq B$ such that for all $c \leq C$, $M_0 \models l(\vec{a},(1,c))$, where $l(\vec{x},y_1)$ is an equational literal of C.

Now consider the P literals of C whose free variables are among the variables $\{x_1,\ldots,x_m,y_1\}$ and that contain an occurrence of y_1. These literals are of the form $\neg P(t(\vec{x},y_1),s^k(y_1))$ and $\neg P(s^k(y_1),t(\vec{x},y_1))$, where t is an $\mathcal{L}_0$ term. For the literals of the form $\neg P(t,s^k(y_1))$ we have $M_0 \models \neg P(t(\vec{a},(1,d)),s^k((1,d)))$, whenever $d < k$. For the literals of the form $\neg P(s^k(y_1),t)$ we have $M_0 \models \neg P(s^k((1,d)),t(\vec{a},(1,d)))$ for all $d \in \mathbb{Z}$. Hence, there is an integer $D \leq B$ such that for all $d \in \mathbb{Z}$ with $d \leq D$, we have $M_0 \models l(\vec{a},(1,d))$, where l is a P literal of C. Now we apply the induction hypothesis in order to obtain suitable valuations for the variables $y_2,\ldots,y_{l+1}$. □

Lemma 6.4.46. $M_0 \models (\mathcal{A}_0 + \mathrm{ACY}) + \exists_1(\mathcal{L}_P)\text{-IND}^{R-}$.

Proof. We proceed by induction on k and to show

$$M_0 \models [\mathcal{A}_0 + \mathrm{ACY}, \exists_1(\mathcal{L}_P)\text{-IND}^{R-}]_k.$$

The base case $k = 0$ is trivial. For the induction step we assume that

$$M_0 \models [\mathcal{A}_0 + \mathrm{ACY}, \exists_1(\mathcal{L}_P)\text{-IND}^{R-}]_k.$$

Now let $\varphi(x)$ be an $\exists_1(\mathcal{L}_P)$ formula, such that φ is inductive in $[\mathcal{A}_0+\mathrm{ACY}, \exists_1(\mathcal{L}_P)\text{-IND}^{R-}]_k$. In particular, we have $M_0 \models \varphi(0)$ and $M_0 \models \varphi(x) \to \varphi(s(x))$. By Lemma 6.4.42 there

exists $N \in \mathbb{N}$ such that $\varphi(s^N(x))$ has over $\mathcal{A}_0 + \mathrm{ACY}$ an equivalent EDNF $\psi(x)$ such that ψ is free of positive equality literals, ground equality literals, and equational literals in one variable. Moreover, $\varphi(s^N(x))$ is also $[\mathcal{A}_0 + \mathrm{ACY}, \exists_1(\mathcal{L}_P)\text{-IND}^{R-}]_k$-inductive. Since $M_0 \models x = 0 \vee (\exists y)x = s(y)$, it suffices to show $M_0 \models \varphi(s^N(x))$. Now let $(i, n) \in \mathcal{D}(M_0)$. The case where $i = 0$ is trivial, since ψ is inductive in M_0 and $M_0 \models (0, n) = \overline{n}$. Now let us consider the case where $i = 1$. By Lemma 6.4.40 we either have $\vdash \psi(x)$, in which case we are done or ψ has a disjunct $(\exists y_1) \dots (\exists y_m) C(x, \vec{y})$ where C is a co-clause that is free of positive P literals and $\bot$. Since C does not contain ground literals we may apply Lemma 6.4.45. Hence, there are integers $b_0, b_1, \dots, b_m$ with $n \geq b_0 \geq b_1 \geq \dots \geq b_m$ such that

$$M_0 \models C((1, b_0), (1, b_1), \dots, (1, b_m)).$$

Hence, $M_0 \models \psi((1, b_0))$ and by a straightforward induction making use of $M_0 \models \psi(x) \to \psi(s(x))$ we obtain $M_0 \models \psi((1, n))$. Thus $M_0 \models [\mathcal{A}_0 + \mathrm{ACY}, \exists_1(\mathcal{L}_P)\text{-IND}^{R-}]_{k+1}$. □

We are now ready to give a proof the main result of this section.

Proof of Theorem 6.4.36. We have $M_0 \models \mathcal{A}_8 + (\varnothing + \exists_1(\mathcal{L}_P)\text{-IND}^{R-})$. □

This solves Conjecture 59 of [HV22]. In particular this result anticipates improvements of clause set cycles by a parameter-free rule like induction mechanism with unrestricted nesting for clause sets.

6.5 Case study: N-clause calculus

The n-clause calculus is a formalism for AITP that was introduced by Kersani and Peltier in [KP13]. The calculus extends a saturation calculus by a mechanism that detects cyclic dependencies between the clauses derived during the saturation process. Such a cyclic dependency represents an argument by infinite descent and, therefore, represents an inductively unsatisfiable subset of the derived clauses. Once such a cycle is detected the refutation is terminated.

In this section we describe a variant of the n-clause calculus [KP13] and show that clause set cycles are a suitable abstraction of the cycles detected by the n-clause calculus. In particular we show that clause set cycles essentially abstract the detection of the cycle but otherwise preserve the argument captured by a cycle.

Definition 6.5.1. *By an n-clause we understand a clause of the form $C(\vec{x}) \leftarrow N(\eta, \vec{x})$ where $N(\eta, \vec{x})$ is a conjunction of literals of the form $\eta = t(\vec{x})$ with t free of η and C is a η-free clause. The formula N is called the constraint part of the n-clause and the clause C is called the clausal part of the n-clause. An n-clause set is a set of n-clauses.*

The original formulation of the n-clause calculus [KP13; Ker14] takes place in a setting that is on the one hand multi-sorted and on the other hand purely equational. As always we work in a one-sorted setting but extending the formalism to a many sorted setting is straightforward. The restriction to a purely equational setting is not necessary for the formulation of the notion of cycles in [KP13] but was adopted in [KP13] because

the underlying superposition calculus operates in a purely equational setting. In [KP13] some other syntactical restrictions are imposed and we omit these in the presentation below because they are not necessary for the formulation of inductive cycles.

The notion of cycles of the n-clause calculus makes use of the descent operator given below to encode the descent step.

Definition 6.5.2. *Let $C = C' \leftarrow N(\eta)$ be an n-clause with $N = \bigwedge_{j=1,\ldots,k} \eta = t_j$. For $i \in \mathbb{N}$ we define $C{\downarrow_i} \coloneqq C(\vec{x}) \leftarrow N(\eta, \vec{x}){\downarrow_i}$ where $N{\downarrow_i} \coloneqq \bigwedge_{j=1,\ldots,k} \eta = s^i(t_j)$. For an n-clause set $\mathcal{C}$ we define $\mathcal{C}{\downarrow_i} \coloneqq \{C{\downarrow_i} \mid C \in \mathcal{C}\}$.*

Intuitively, the $\downarrow_j$ operation allows us to express that η is replaced by its j-th predecessor. The following lemma states a crucial property of the $\downarrow_j$ operator.

Lemma 6.5.3. *Let $S(\eta)$ be clause set and $j \geq 0$, then we have $S{\downarrow_j}(s^j\eta) \models S(\eta)$.*

Proof. Straightforward. □

The converse of the above entailment holds when the injectivity of the successor function is provided.

Lemma 6.5.4. *Let $S(\eta)$ be a clause set and $j \geq 0$, then*

$$(\forall x)(\forall y)(s(x) = s(y) \rightarrow x = y) + S(\eta) \models S{\downarrow_j}(s^j\eta).$$

Proof. Let $C \leftarrow N \in S$ with $N = \bigwedge_{i=1}^{k} \eta = t_i$ and assume $N{\downarrow_j}(s^j(\eta))$, that is, $\bigwedge_{i=1}^{k} s^j(\eta) = s^j(t_i)$. By injectivity of s we obtain $\bigwedge_{i=1} \eta = t_i$. Thus we obtain C. □

We can now introduce the notions of cycle and of refutability by a cycle.

Definition 6.5.5. *Let $\mathcal{D}(\eta)$ be an n-clause set. A triple $(i, j, \mathcal{C}(\eta))$ with $i, j \in \mathbb{N}$, $j > 0$ and $\mathcal{C} \subseteq \mathcal{D}$ is a cycle for $\mathcal{D}$ if $\mathcal{C} \models \eta \neq \overline{k}$ for $k = i, \ldots, i + j - 1$ and $\mathcal{C} \models \mathcal{C}{\downarrow_j}$. We say that $\mathcal{D}$ is refuted by a cycle if there exists a cycle $(i, j, \mathcal{C})$ for $\mathcal{D}$ and $\mathcal{D} \models \eta \neq \overline{k}$, for $k = 0, \ldots, i - 1$.*

As already mentioned above we consider a simplified variant of the n-clause calculus defined in [KP13]. Only one of the simplifications imposed by us restricts the power of the formalism. In [KP13] the entailments rely on semantics that interpret the sort of natural numbers as the natural numbers. Instead our definition above works with the usual first-order semantics and thus gives us access to the completeness theorem for first-order logic. Clearly, this restricts the overall power of the formalism but for our purposes this is inessential, since we are interested in effective systems. In particular, the practical variants of inductive cycles given in [KP13] use a decidable entailment checks such as the entailment relation generated by the superposition calculus, subsumption or syntactical equality up to renaming of variables that imply first-order entailment. Hence our restriction does not rule out any practically relevant instances of the n-clause calculus.

A cycle $(i, j, \mathcal{C}(\eta))$ for a clause set $\mathcal{D}(\eta)$ is similar to an argument by induction with an external offset i and a descent step size j. Accordingly, the conditions $\mathcal{C} \vdash n \neq \overline{k}$

for $k = i, \ldots, i + j - 1$ correspond to the j base cases, whereas the condition $\mathcal{C} \vdash \mathcal{C}{\downarrow_j}$ corresponds to the step case.

Proposition 6.5.6. *Let $\mathcal{D}$ be an n-clause set refuted by a cycle, then $\mathcal{D}$ is refuted by a $\mathcal{L}(\mathcal{D}) \setminus \{\eta\}$ clause set cycle.*

Proof. Let $(i, j, \mathcal{S}(\eta))$ be a cycle refuting $\mathcal{D}$. Moreover, let $\mathcal{C}(\eta)$ be the clause set $\mathcal{S}(s^i(\eta))$. We will show that $\mathcal{C}(\eta)$ is a $(0, j)$-clause set cycle and that $\mathcal{D}$ is refuted by $\mathcal{C}$ with external offset i. First of all, observe that since $\mathcal{S} \models \eta \neq \overline{k}$ for $k = i, i + 1, \ldots, i + j - 1$, we have $\mathcal{C}(\overline{k}) \models \bot$ for $k = 0, \ldots, j - 1$. Furthermore, since $\mathcal{S} \models \mathcal{S}{\downarrow_j}$, we have $\mathcal{S}(s^{i+j}(\eta)) \models \mathcal{S}{\downarrow_j}(s^{i+j}(\eta))$. Hence, by the definition of $\mathcal{C}$ and by Lemma 6.5.3, we obtain $\mathcal{C}(s^j(\eta)) \models \mathcal{C}(\eta)$. Thus $\mathcal{C}$ is a $(0, j)$-clause set cycle. Now observe that since $\mathcal{S} \subseteq \mathcal{D}$, we have $\mathcal{D}(\eta) \models \mathcal{S}(\eta)$ and therefore $\mathcal{D}(s^i\eta) \models \mathcal{C}(\eta)$. Finally, $\mathcal{D} \models \eta \neq \overline{k}$, hence $\mathcal{D}(\overline{l}) \models \bot$ for $l = 0, \ldots, i - 1$. Thus $\mathcal{D}$ is refuted by the $(0, j)$-clause set cycle with external offset i. Hence, by Proposition 6.1.10 $\mathcal{D}$ is refuted by a clause set cycle. □

Thus, this shows us that refutation by a clause set cycles are an abstraction of refutation by a cycle of the n-clause calculus. Therefore, all the unprovability results developed in Section 6.4 apply to the n-clause calculus.

In Definition 6.5.5, the condition that the n-clause set $\mathcal{C}(\eta)$ is a subset of $\mathcal{D}(\eta)$ is an analyticity condition that is analogous to the detection of clause set cycles by the rule $\mathrm{CSC_S}$ in the set of clauses derived by the underlying saturation system (see Definition 6.1.5 and the proof of Lemma 6.1.6). Because of this condition systems based cycles of the n-clause calculus may behave differently than systems based on the rule $\mathrm{CSC_S}$. However, by weakening this condition to $\mathcal{D} \models \mathcal{C}$ it can be seen that a refutation by a clause set cycle can be turned into a refutation by a cycle of the n-clause calculus.

Lemma 6.5.7. *Let $\mathcal{L}$ be a language not containing η. Let $\mathcal{D}$ be an $\mathcal{L} \cup \{\eta\}$ clause set such that $\mathcal{D}$ is refuted by a clause set cycle. Then there exists an n-clause set $\mathcal{C}(\eta)$ over the language $\mathcal{L} \cup \{\eta\}$ such that $\mathcal{C} \models \eta \neq 0$, $\mathcal{C} \models \mathcal{C}{\downarrow_1}$ and $\mathcal{D} \models \mathcal{C}$.*

Proof. Let $\mathcal{C}'$ be a clause set cycle that refutes $\mathcal{D}$. Now let $\mathcal{C}$ the n-clause set

$$\mathcal{C}'[\eta/x] \leftarrow \eta = x.$$

It is obvious that $\mathcal{C}$ is logically equivalent to $\mathcal{C}'$. Hence, since $\mathcal{C}'[\eta/0] \models \bot$, we have in particular $\mathcal{C} \models \eta \neq 0$. Now let us show that $\mathcal{C} \models \mathcal{C}{\downarrow_1}$. Let M be a structure such that $M \models \mathcal{C}$ and let $d \in \mathcal{D}(M)$ such that $M \models \eta = s(d)$. We have to show $M, \{x \mapsto d\} \models \mathcal{C}'[\eta/x]$. By the assumption $M \models \mathcal{C}$, we have $M, \{x \mapsto s(d)\} \models \mathcal{C}'[\eta/x]$. Now let $N = M[\eta \mapsto d]$, then $N \models \mathcal{C}'[\eta/s(\eta)]$. Since $\mathcal{C}'$ is a clause set cycle, we have $N \models \mathcal{C}'$. Therefore, by the construction of N we also have $M, \{x \mapsto d\} \models \mathcal{C}'[\eta/x]$. Finally, observe that $\mathcal{D} \models \mathcal{C}$. □

This shows that refutation by a clause set cycle is a simpler reformulation of the cycles detected by the n-clause calculus that abstracts the analyticity of the cycle detection. In other words, by lifting the analyticity inherent to the cycles of the n-clause calculus we obtain a formalism that is exactly as strong as clause set cycles.

6.6 Summary

In this chapter we have considered refutation by clause set cycles. Clause set cycles are a form of inductive dependency between clause sets that are an abstraction of the cycles detected by the n-clause calculus [KP13; Ker14]. We have shown that refutation by a clause set cycle corresponds essentially to unnested applications of the parameter-free induction rule over $\exists_1$ formulas (Theorem 6.3.6). This observation has allowed us to provide various simple properties that cannot be refuted with clause set cycles but that can be refuted by relatively straightforward arguments by induction. On the other hand we have also seen that clause set cycles permit us to prove properties that are not even provable by comparatively powerful calculi such as the calculus considered in Chapter 5.

The results presented in this thesis allow us to locate with respect to other systems with induction. These situation is summarized in Fig. 6.1. The figure quantifies over all languages not containing the symbol η and shows the relative refutational strength for various systems with induction. A system is represented by an arc and the name of the system is near the top of the arc. The system $\_ + \mathrm{CSC_S}$ is particular in that it stands for the analytic family of sound saturation systems extended the clause set cycle detection rule (see Definition 6.1.5) that are guaranteed to detect clause set cycles present in the initial clause sets. Since these systems may vary in strength depending on the underlying sound saturation system this family is delimited by a dashed arc. The points $\{\bullet_i \mid i = 1, \ldots, 6\}$ correspond to clause sets whose position is confirmed by the results in this thesis and [HV20; HV22]. In particular, the point $\bullet_1$ is witnessed for instance by the clause set considered in Lemma 6.2.1. The point $\bullet_2$ corresponds to the clause set considered in Section 6.4.1. The point $\bullet_3$ corresponds to the clause set considered in Section 6.4.2. Furthermore, the points $\bullet_4$ and $\bullet_5$ correspond to some of the clause sets considered in Section 6.4.3. Finally, the point $\bullet_6$ corresponds to the clause set constructed in Section 6.4.4. The inclusion of the system Open($\mathcal{L}$)-IND$^-$ in the system $[\varnothing, \exists_1(\mathcal{L})\text{-IND}^{R-}]$ is shown by Lemma 6.3.11.

Despite the many limitations of clause set cycles shown in this thesis, refutation by a clause set cycle is an interesting way of integrating induction into saturation-based provers, in particular because the detection of clause set cycles does not generate any additional clauses, except perhaps an empty clause, and therefore neither swamps the prover with clauses nor extends the working language. This suggests that methods that integrate induction in terms of Hilbert style induction rules could be more efficient, albeit slightly weaker, alternatives for the methods discussed in Chapters 4 and 5. It may be possible to detect inductive dependencies corresponding to the premises of the Hilbert-style induction rule between clauses sets using similar techniques as used in [KP13]. Finally, let us mention a possible extension of the formalism of refutation by a clause set cycle. The detection of clause set cycles might be combined with a case split rule that allows us to split a clause set $\mathcal{C}(c)$ where c is a constant into two new clause sets $\mathcal{C}(0)$ and $\mathcal{C}(s(c))$ that are to be refuted. Now cyclic dependencies could be detected between the generated clause sets resulting in a refutational variant of the cyclic systems described in [BS11] and implemented in [BGP12]. Possibly such a calculus could make use of existing clause splitting architectures.

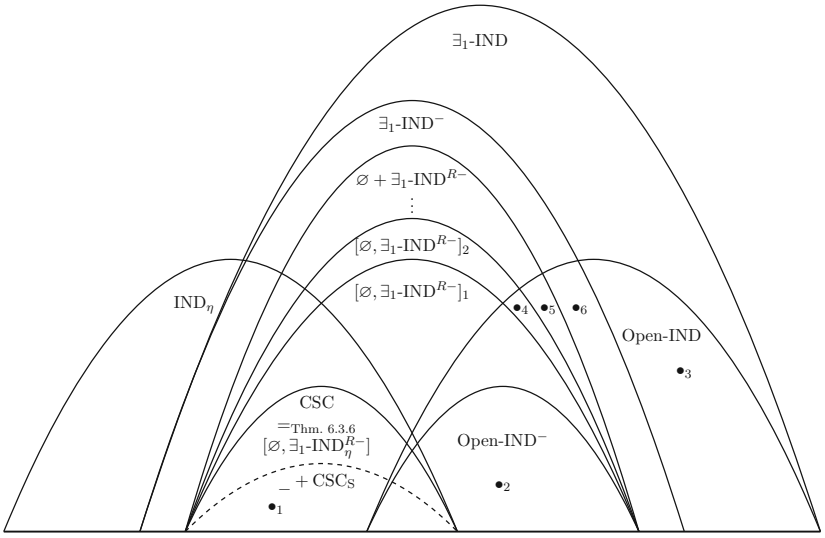

Figure 6.1: Overview of the clausal refutational strength of various induction systems.

7 Cancellation in linear arithmetic

In this chapter we will provide a proof for Theorem 6.4.8 which states that over the base theory of linear arithmetic the parameter-free induction schema for $\exists_1$ formulas is not strong enough to prove some univariate cancellation properties of addition. This independence result is interesting for AITP because it shows that certain properties cannot be proved without introducing additional variables. This result has been used in particular in Section 6.4.2 to provide simple unprovability results for refutation by a clause set cycle. In Section 7.1 we introduce some preliminary notions and we carry out some syntactic simplifications on $\exists_1$ formulas. In Section 7.2 we consider some properties of $\exists_1$ formulas over natural numbers and integers. After that, in Section 7.3 we construct models for the parameter-free $\exists_1$ induction rule. These models allow us to show that over the base theory of linear arithmetic the parameter-free $\exists_1$ induction rule is strictly weaker than the corresponding induction schema. This shows us in particular that Theorem 6.4.8 is strictly stronger than [HV22, Theorem 53]. Furthermore, this gives rise to the interesting question whether the system based on the induction schema is conservative for $\forall_2$ sentences over the system based on the induction rule. Finally, in Section 7.4 we construct a model for the parameter-free $\exists_1$ induction schema and give a proof of Theorem 6.4.8.

7.1 Preliminaries

In this section we mainly carry out some syntactic transformations that allow us to eliminate the function symbols p and 0 from $\exists_1$ formulas. The absence of these symbols allows us to carry out certain embeddings of structures in Sections 7.2 and 7.3. We work in the setting of linear arithmetic, hence, unless stated otherwise, whenever we speak of a formula (sentence) we mean an $\mathcal{L}_{\mathrm{LA}}$ formula (sentence).

Definition 7.1.1. *The theory $\mathcal{V}$ is axiomatized by the universal closure of the formulas*

$$\overline{k} + x = x + \overline{k}, \tag{$\mathrm{V_k}$}$$

where $k \in \mathbb{N}$.

Lemma 7.1.2. $[B, \mathrm{Open}(\{0, s, +\})\text{-IND}^{R-}] \vdash \mathcal{V}$.

Proof. The formula $\overline{k} + x = x + \overline{k}$ is B-inductive. □

We will carry out these transformations in the very weak theory $B + \mathrm{B1} + \mathcal{V}$. In a first step we will show that we can eliminate the symbol p from $\exists_1$ formulas without

increasing the quantifier complexity of $\exists_1$ formulas. After that, we show that we can moreover eliminate to a certain extent the symbol 0 from $\exists_1$ formulas, again without increasing the quantifier complexity.

In order to eliminate the symbol p from $\exists_1$ formulas we proceed by replacing all the occurrences of the symbol p by the following definition of the predecessor function.

Definition 7.1.3. *We define the formula* $\mathrm{D}(x, y)$ *by*

$$(x = 0 \wedge y = 0) \vee s(y) = x.$$

Lemma 7.1.4. $B + \mathrm{B1} \vdash p(x) = y \leftrightarrow \mathrm{D}(x, y)$.

Proof. We work in B+B1. Assume $p(x) = y$. If $x = 0$, then we have $y = p(x) = p(0) = 0$, hence $\mathrm{D}(x, y)$. Otherwise, $x = s(p(x))$ and therefore $x = s(p(x)) = s(y)$. Now assume $\mathrm{D}(x, y)$. If $x = 0 \wedge y = 0$, then we have $p(x) = p(0) = 0 = y$. If $s(y) = x$, then $y = p(s(y)) = p(x)$. □

We can now factor the symbol p into the axiom B1 by replacing all the occurrences of p by the definition of the predecessor function.

Lemma 7.1.5. *Let* $\varphi(\vec{x})$ *be an* $\exists_1$ $\mathcal{L}_{\mathrm{LA}}$ *formula, then there exists a* p*-free* $\exists_1$ *formula* $\varphi'(\vec{x})$ *such that*

$$B + \mathrm{B1} \vdash \varphi \leftrightarrow \varphi'.$$

Proof. Let φ be an $\exists_1(\mathcal{L}_{\mathrm{LA}})$ formula, then there exists an unnested $\exists_1(\mathcal{L}_{\mathrm{LA}})$ formula ψ such that $\vdash \varphi \leftrightarrow \psi$, see for example [Hod97, pp. 51–52]. In particular, the symbol p occurs in ψ only in atoms of the form $p(x) = y$. Hence, we obtain the desired formula by replacing in ψ the atomic formulas of the form $p(x) = y$ by $\mathrm{D}(x, y)$. □

In the following we will eliminate the symbol 0 to a certain extent from $\exists_1$ formulas in one variable. In order to simplify the arguments we will introduce some additional assumptions. Since we work in the context of the theory B we can by Lemma 2.6.10 assume without loss of generality that ground terms are numerals. Moreover, since equality is symmetric we will assume without loss of generality that atoms are oriented in such a way that whenever the atom contains a variable, then the left hand side of the atom contains a variable.

Lemma 7.1.6. *Let* $\varphi(x)$ *be an* $\exists_1$ *formula, then there exist* p*-free components*[1] $\chi_1, \ldots, \chi_n$ *such that* $B + \mathrm{B1} \vdash \varphi \leftrightarrow \bigvee_{i=1}^{n} \chi_i$.

Proof. Apply Lemma 7.1.5 to obtain a p-free $\exists_1$ formula φ' such that $B + \mathrm{B1} \vdash \varphi \leftrightarrow \varphi'$. Now obtain the desired components by replacing formulas of the form $\varphi \to \psi$ and $\varphi \leftrightarrow \psi$ respectively by $\neg\varphi \vee \psi$ and $(\neg\varphi \vee \psi) \wedge (\neg\psi \vee \varphi)$, moving negations inward, eliminating double negations, distributing conjunctions over disjunctions, and moving existential quantifiers inwards over disjunctions. □

[1] See Definition 6.4.39

We will distinguish between three types of literals: Those where both sides contain variables, those where only one side of the equation contains a variable and those where none of the sides contain a variable.

Definition 7.1.7. *Let l be a literal of the form $u \bowtie v$ with $\bowtie \in \{=, \neq\}$, then l is: $\uparrow\uparrow$ if both u and v contain a variable, $\uparrow\downarrow$ if u contains a variable and v is ground, and $\downarrow\downarrow$ if both u and v are ground. We will combine this notation with superscript $+$ to indicate that the literal is positive and a superscript $-$ to indicate that the literal is negative. We say that a $\uparrow\downarrow$ literal is simple if it is of the form $z \bowtie \overline{k}$ where $\bowtie \in \{=, \neq\}$, z is a variable and $k \in \mathbb{N}$. An $\uparrow\downarrow$ literal is complex if it is not simple.*

Lemma 7.1.8. *Let t be a term with $Var(t) \neq \varnothing$, then there exists a 0-free term t' such that $B + \mathcal{V} \vdash t = t'$.*

Proof. We proceed by induction on the structure of the term t. If t is a variable, then we are done by letting $t' = t$. If t is of the form $s(u)$, then $Var(u) \neq \varnothing$. Hence, we can apply the induction hypothesis to u in order to obtain a 0-free term u' such that $B + \mathcal{V} \vdash u = u'$. Thus, $B + \mathcal{V} \vdash t = s(u')$ and we let $t' = s(u')$. If t is of the form $p(u)$, then we proceed analogously. If t is of the form $u + v$, then we need to consider several cases. If $Var(u) = \varnothing$, then $Var(v) \neq \varnothing$ and we have $B \vdash u = \overline{k}$ for some $k \in \mathbb{N}$ and therefore $B + \mathcal{V} \vdash t = \overline{k} + v = v + \overline{k} = v' + \overline{k} = s^k(v')$. If $Var(v) = \varnothing$, then $Var(u) \neq \varnothing$. Hence, we can apply the induction hypothesis to u in order to obtain a 0-free term u' such that $B + \mathcal{V} \vdash u = u'$. Since $Var(v) = \varnothing$, there exists $k \in \mathbb{N}$ such that $B \vdash v = \overline{k}$. By multiple applications of (A5) followed by an application of (A5) we obtain $B + \mathcal{V} \vdash t = u + \overline{k} = s^k(u) + 0 = s^k(u)$. Hence, $t' \coloneqq s^k(u)$ is the desired 0-free term. If u and v contain variables, then by the induction hypothesis we obtain 0-free terms u' and v' such that $B + \mathcal{V} \vdash u = u'$ and $B + \mathcal{V} \vdash v = v'$. Hence, $t = u' + v'$ is the desired 0-free term. □

By Lemma 2.6.11 and Lemma 7.1.8, it is straightforward to eliminate the symbol 0 from $\uparrow\uparrow$ and $\downarrow\downarrow$ literals. However, eliminating the symbol 0 from $\uparrow\downarrow$ literals needs some more work. Let us start by observing that complex $\uparrow\downarrow$ atoms can be split into several simple ones.

Lemma 7.1.9. *Let $u(\vec{z})$ be a p-free term with $\vec{z} = (z_1, \ldots, z_l)$ and $k \in \mathbb{N}$, then*

$$B + \mathrm{B1} \vdash u(\vec{z}) = \overline{k} \leftrightarrow \bigvee_{\substack{0 \leq m_1, \ldots, m_l \leq k \\ u^{\mathbb{N}}(m_1, \ldots, m_l) = k}} \bigwedge_{j=1}^{l} z_j = \overline{m_j}.$$

Proof. Work in $B + \mathrm{B1}$. The "if" direction is obvious. For the "only if" direction assume $u(\vec{z}) = \overline{k}$ and proceed by k-fold case analysis on the variables $\vec{z}$. If $z_i = \overline{m_i}$ with $0 \leq m_i \leq k$ for $i = 1, \ldots, l$, then we have two cases. If $u(\overline{m_1}, \ldots, \overline{m_l}) \neq \overline{k}$, then the claim follows trivially. Otherwise if $u(\overline{m_1}, \ldots, \overline{m_l}) = \overline{k}$, then we are done as well since $z_1 = \overline{m_1} \wedge \cdots \wedge z_l = \overline{m_l}$ is a conjunct of the right side. Otherwise, there exists an $i \in \{1, \ldots, l\}$ and z_i' such that $z_i = s^{k+1} z_i'$. Then let j be the index of the variable

z_j with the rightmost occurrence such that $z_j = s^{k+1}(z'_j)$ for some z'_j. Then we have $u(\vec{z}) = s^{k+1}(u'(z_1, \ldots, z_{j-1}, z'_j, z_{j+1}, \ldots, z_l))$ and a term u'. Hence, by Lemma 2.6.9.(i) we have $u(\vec{z}) \neq \overline{k}$. □

Furthermore, we can eliminate simple $\uparrow\downarrow^-$ literals at the expense of introducing several positive literals and an existential quantifier.

Lemma 7.1.10. *Let $k \in \mathbb{N}$, then*

$$B + \mathrm{B1} \vdash z \neq \overline{k} \leftrightarrow \left(\exists z' z = s^{k+1} z' \vee \bigvee_{i=0}^{k-1} z = \overline{i} \right).$$

Proof. The "if" direction is obvious. For the "only if" direction assume $z \neq \overline{k}$ and proceed by k-fold case analysis on z. If $z = \overline{i}$ with $i < k$, then we are done. The case where $z = \overline{k}$ contradicts the assumption and therefore we are done. If $z = s^{k+1}z'$ for some z', then we are done as well. □

The elimination of the $\uparrow\downarrow$ literals from a component $\chi(x)$ consists of two majors steps. In a first step we deal with all the $\uparrow\downarrow$ literals except the simple $\uparrow\downarrow$ literals containing the variable x. In the second step we will deal with the remaining $\uparrow\downarrow$ literals by making use of the observation that the truth value of these literals becomes fixed when x is large enough.

Let us start by defining some measures that will be used to control the first step of the elimination procedure.

Definition 7.1.11. *Let $\chi(\vec{x}) = (\exists y_1) \ldots (\exists y_l) C_\chi$ be a component, then $\#^-(\chi)$ is the number of occurrences of negative literals in χ, $\#_\exists(\chi) = l$, $\#^+_{\mathrm{complex}}(\chi)$ is the number of occurrences of complex $\uparrow\downarrow^+$ literals in χ, and $\#_{\mathrm{FV}}(\chi)$ is the number of free variables of the component χ.*

We will now provide some intermediate lemmas that allow us to eliminate a single literal.

Lemma 7.1.12 (Elimination of $\uparrow\downarrow^-$ literals)**.** *Let $\chi(x)$ be a p-free component containing a $\uparrow\downarrow^-$ literal, then there exist p-free components $\chi'_1, \ldots, \chi'_n$ such that B+B1 $\vdash \chi \leftrightarrow \bigvee_{i=1}^n \chi'_i$ and $\#^-(\chi'_i) < \#^-(\chi)$ for $i = 1, \ldots, n$.*

Proof. We first apply Lemma 7.1.9 in order to split the atom of the $\uparrow\downarrow^-$ literal. After that, we move the negations inwards and apply Lemma 7.1.10 to all the newly introduced literals of the form $z \neq \overline{k}$ with $k \in \mathbb{N}$. Now we move the newly introduced existential quantifiers outwards and possibly rename some bound variables. Finally, we distribute conjunctions over disjunctions exhaustively. Let $\chi_1, \ldots, \chi_k$ be the resulting components. Since we have introduced only existential quantifiers and positive literals, we have $\#^-(\chi_i) < \#^-(\chi)$. □

Lemma 7.1.13 (Elimination of complex $\uparrow\downarrow^+$ literals)**.** *Let $\chi(x)$ be a p-free component containing a complex $\uparrow\downarrow^+$ literal, then there exist p-free components $\chi'_1, \ldots, \chi'_n$ with $B + \mathrm{B1} \vdash \chi \leftrightarrow \bigvee_{i=1}^n \chi'_i$ such that $\#^-(\chi'_i) = \#^-(\chi)$, $\#_\exists(\chi'_i) = \#_\exists(\chi)$, and $\#^+_{\mathrm{complex}}(\chi'_i) < \#^+_{\mathrm{complex}}(\chi)$ for $i = 1, \ldots, n$.*

Proof. We apply Lemma 7.1.9 to split a complex $\uparrow\downarrow^+$ literal. Now obtain components $\chi'_1, \ldots, \chi'_n$ by distributing conjunctions over disjunctions exhaustively and moving the existential quantifiers inwards over the disjunctions. Clearly we have $B + \mathrm{B1} \vdash \chi \leftrightarrow \bigvee_{i=1}^n \chi'_i$. Observe that this operation does not introduce any negative literals or quantifiers. Hence we have $\#^-(\chi'_i) = \#(\chi)$ and $\#_\exists(\chi'_i) = \#_\exists(\chi)$ for $i = 1, \ldots, n$. Moreover, only simple $\uparrow\downarrow^+$ literals are introduced, hence we have $\#^+_{\mathrm{complex}}(\chi'_i) < \#^+_{\mathrm{complex}}(\chi)$ for $i = 1, \ldots, n$. □

Lemma 7.1.14 (Elimination of simple $\uparrow\downarrow^+$ literals)**.** *Let $\chi(x) = \exists y_1, \ldots, y_l C$ be a p-free component containing a literal of the form $y_i = \overline{k}$, then there exists a p-free component $\chi'(x)$ such that $\vdash \chi \leftrightarrow \chi'$, $\#^-(\chi') = \#^-(\chi)$, and $\#_\exists(\chi') < \#_\exists(\chi)$.*

Proof. Let us assume without loss of generality that $C = y_i = \overline{k} \wedge C'(y_1, \ldots, y_l)$, where C' is a conjunction of literals. Then, it suffices to apply the first-order equivalence

$$\vdash \exists y_i \left(y_i = \overline{k} \wedge C'(x, y_1, \ldots, y_{i-1}, y_i, y_{i+1}, \ldots, y_l)\right) \leftrightarrow C'(y_1, \ldots, y_{i-1}, \overline{k}, y_{i+1}, \ldots, y_l).$$

Clearly we have $\#^-(\chi') = \#^-(\chi)$ and $\#_\exists(\chi') < \#_\exists(\chi)$. □

The following lemma combines the previous lemmas in order to accomplish the first step of the elimination of the $\uparrow\downarrow$ literals.

Lemma 7.1.15. *Over $B + \mathrm{B1} + \mathcal{V}$ every $\exists_1(\mathcal{L}_{\mathrm{LA}})$ formula $\varphi(x_1, \ldots, x_n)$ is equivalent to a disjunction of formulas of the form $\bigwedge_{i \in I} x_i = \overline{k_i} \wedge (\exists \vec{y}) C(\vec{x}, \vec{y})$, where $I \subseteq [n] = \{1, \ldots, n\}$ and C is a p-free 0-free conjunction of literals that contains only those variables x_i such that $i \notin I$.*

Proof. Let $\chi(\vec{x})$ be a p-free component, then we proceed by induction on the lexicographic order $\prec$ on $\mathbb{N}^4$ induced by $\leq$ and show that over $B + \mathrm{B1}$ the component χ is equivalent to disjunction of formulas of the form $\bigwedge_{i \in I} x_i = \overline{k_i} \wedge (\exists \vec{y}) C(\vec{x}, \vec{y})$, where $I \subseteq [n]$ and C is a p-free disjunction of $\uparrow\uparrow$ and $\downarrow\downarrow$ literals that contains only those variables x_i such that $i \notin I$. Let $\#(\chi) = (\#^-(\chi), \#_\exists(\chi), \#^+_{\mathrm{complex}}(\chi), \#_{\mathrm{FV}}(\chi))$. If χ contains a $\uparrow\downarrow^-$ literal, then we apply Lemma 7.1.12 in order to obtain p-free components $\chi'_1, \ldots, \chi'_n$ such that $B + \mathrm{B1} \vdash \chi \leftrightarrow \bigvee_{i=1}^n \chi'_i$ and $\#^-(\chi'_i) < \#^-(\chi)$. Hence $\#(\chi'_i) \prec \#(\chi)$ and therefore we can apply the induction hypothesis to each of $\chi'_1, \ldots, \chi'_n$ in order to obtain the desired components. If χ contains a complex $\uparrow\downarrow^+$ literal, then we apply Lemma 7.1.13 in order to obtain p-free components $\chi'_1, \ldots, \chi'_n$ with $B + \mathrm{B1} \vdash \chi \leftrightarrow \bigvee_{i=1}^n \chi'_i$ such that $\#^-(\chi'_i) = \#^-(\chi)$, $\#_\exists(\chi'_i) = \#_\exists(\chi)$ and $\#^+_{\mathrm{complex}}(\chi'_i) < \#^+_{\mathrm{complex}}(\chi)$, for $i = 1, \ldots, n$. Hence $\#(\chi'_i) \prec \#(\chi)$ for $i = 1, \ldots, n$ and therefore we can apply the induction hypothesis to $\chi'_1, \ldots, \chi'_n$ in order to obtain the desired components. Let $\chi(x) = (\exists y_1) \ldots (\exists y_l) C_\chi$. If

χ contains a $\uparrow\downarrow$ literal of the form $x_i = \overline{k_i}$ with $i \in \{1, \ldots, n\}$, then let $\chi = (\exists \vec{y}) C_\chi(\vec{x}, \vec{y})$ and $\chi' = (\exists \vec{y}) C_\chi[x_i/\overline{k_i}]$. We have $\vdash \chi \leftrightarrow x_i = \overline{k_i} \wedge \chi'$. Clearly, $\#^-(\chi') = \#^-(\chi)$, $\#_\exists(\chi') = \#_\exists(\chi)$, and $\#^+_{\text{complex}}(\chi') \leq \#^+_{\text{complex}}(\chi)$ but $\#_{\text{FV}}(\chi') = \#_{\text{FV}}(\chi) - 1$. Hence, we may apply the induction hypothesis to the component χ'. If χ contains a simple $\uparrow\downarrow^+$ literal $y_i = \overline{k}$, then we apply Lemma 7.1.14 in order to obtain a p-free component $\chi'(x)$ such that $B + \text{B1} \vdash \chi \leftrightarrow \chi'$, $\#^-(\chi') = \#^-(\chi)$, and $\#_\exists(\chi') < \#_\exists(\chi)$. Hence we have $\#(\chi') \prec \#(\chi)$ and therefore we can apply the induction hypothesis in order to obtain the desired components.

Now let $\varphi(x_1, \ldots, x_n)$ be an $\exists_1(\mathcal{L}_{\text{LA}})$ formula. By Lemma 7.1.6 the formula φ is equivalent over B+B1 to a disjunction of p-free components. Therefore, by the procedure above the formula φ is equivalent over $B + \text{B1}$ to a disjunction of formulas of the form $\bigwedge_{i \in I} x_i = \overline{k_i} \wedge (\exists \vec{y}) C(\vec{x}, \vec{y})$, where $I \subseteq [n]$ and C is a p-free disjunction of $\uparrow\uparrow$ and $\downarrow\downarrow$ literals containing only those variables x_i such that $i \notin I$. Now we apply Lemma 2.6.11 to eliminate the $\downarrow\downarrow$ literals from C and Lemma 7.1.8 to eliminate 0 from the $\uparrow\uparrow$ literals of C. □

In the next step we eliminate the remaining literals of the form $x = \overline{k}$. This step relies on the observation that the truth value of these literals is fixed when x is large enough.

Proposition 7.1.16. *Let $\varphi(x_1, \ldots, x_n)$ be an $\exists_1$ formula, then there exists $N \in \mathbb{N}$ such that $\varphi(s^N(x_1), \ldots, s^N(x_n))$ is equivalent over $B + \text{B1} + \mathcal{V}$ to a 0-free, p-free, $\exists_1$ formula.*

Proof. By Lemma 7.1.15 the formula φ is equivalent over $B + \text{B1} + \mathcal{V}$ to a disjunction of the form

$$\bigvee_{j=1}^{m} \left(\bigwedge_{i \in I_j} x_i = \overline{k}_{j,i} \wedge (\exists \vec{y}) C_j(\vec{x}, \vec{y}) \right),$$

where for $j = 1, \ldots, m$, $I_j \subseteq [n]$ and C_j is a p-free 0-free disjunction of literals containing only those variables x_i such that $i \notin I_j$. Let $N = 1 + \max\{k_{i,j} \mid j = 1, \ldots, m, i \in I_j\}$, then $\varphi(s^N(x_1), \ldots, s^N(x_n))$ is equivalent over $B + \text{B1} + \mathcal{V}$ to the formula

$$\bigvee_{\substack{j=1,\ldots,m \\ I_j = \varnothing}} (\exists \vec{y}) C_j(\vec{x}, \vec{y}).$$

Finally, we obtain the desired formula by moving the $\exists$ quantifiers outwards over the disjunction. □

7.2 Components in $\mathbb{N}$ and $\mathbb{Z}$

In this section we will investigate some basic model-theoretic properties of $\exists_1$ formulas over natural numbers and integers.

We denote by $\mathbb{Z}$ the set of integers as well as the $\mathcal{L}_{\text{LA}}$ structure whose domain is the set of integers and that interprets all symbols naturally. In particular, $\mathbb{Z}$ interprets the symbol p as the function $x \mapsto x - 1$. Clearly, we have $\mathbb{Z} \models \text{A1} + \text{A3} + \text{A4} + \text{A5}$, but $\mathbb{Z} \not\models \text{A2}$, since $\mathbb{Z} \models p(0) = -1$.

Definition 7.2.1. *Let M be an $\mathcal{L}_{\mathrm{LA}}$ structure and $\varphi(\vec{x})$ a formula. We say that $\vec{d} \in \mathcal{D}(M)^{|\vec{x}|}$ is a solution of φ in M if $M \models \varphi(\vec{d})$.*

We will show that every p-free $\exists_1$ formula with enough solutions in $\mathbb{N}|_{\{0,s,+\}}$ has an infinite strictly descending sequence of solutions in $\mathbb{Z}|_{\{0,s,+\}}$. Let $\varphi(x)$ be an $\exists_1$ formula, then since the structure $\mathbb{N}|_{\{0,s,+\}}$ can be embedded into $\mathbb{Z}|_{\{0,s,+\}}$, it is clear that $\mathbb{N}|_{\{0,s,+\}} \models \varphi(n)$ implies $\mathbb{Z}|_{\{0,s,+\}} \models \varphi(n)$, for all $n \in \mathbb{N}$.

Let $\theta(x_1, \ldots, x_k)$ be an atom, then it is obvious that θ is equivalent in $\mathbb{Z}$ to a linear equation of the form $\sum_{i=1}^{k} a_i x_i = b$ with integers $a_1, \ldots, a_k, b$. Hence a conjunction of atoms $\theta_1(x_1, \ldots, x_k), \ldots, \theta_n(x_1, \ldots, x_k)$ is equivalent over $\mathbb{Z}$ to an inhomogeneous system of linear equations of the form

$$A\vec{x} = b, \tag{I}$$

where $A \in \mathbb{Z}^{m \times k}$ and $b \in \mathbb{Z}^{m \times 1}$. Now consider the corresponding homogeneous system

$$A\vec{x} = \mathbf{0}. \tag{H}$$

The solutions of the system (H) form a submonoid $\mathcal{H}$ of $\mathbb{Z}^k$ with pointwise addition. Furthermore, assume that (I) has a particular solution $i_{(p)}$, then the set of solutions of (I) is given by

$$\mathcal{I} = \{h + i_{(p)} \mid h \in \mathcal{H}\}.$$

Lemma 7.2.2. *Let $C(\vec{x})$ be a conjunction of $\mathcal{L}_{\mathrm{LA}}$ atoms and let $\vec{m}, \vec{n}$ be solutions in $\mathbb{Z}$ of C, then $\mathbb{Z} \models C(k(\vec{m} - \vec{n}) + \vec{m})$ for all $k \in \mathbb{Z}$.*

Proof. As mentioned above the conjunction of atoms $C(\vec{x})$ with $|\vec{x}| = p$ is equivalent in $\mathbb{Z}$ to an inhomogeneous linear system

$$A\vec{x}^T = b, \tag{I}$$

with $A \in \mathbb{Z}^{l \times p}$ and $b \in \mathbb{Z}^{l \times 1}$, where l is the number of conjuncts of C. Let us denote by $\mathcal{I}$ the set of solutions of (I) and by $\mathcal{H}$ the set of solutions of the homogeneous system. Then $\vec{m}^T, \vec{n}^T \in \mathcal{I}$ and therefore $h_0 \coloneqq \vec{m} - \vec{n} \in \mathcal{H}$. Hence $k \cdot h_0 + \vec{m} \in \mathcal{I}$ for all $k \in \mathbb{N}$. □

Lemma 7.2.3. *Let $C(\vec{x})$ be a conjunction of negative $\mathcal{L}_{\mathrm{LA}}$ literals and let $\vec{m}, \vec{n}$ be solutions in $\mathbb{Z}$ of C, then there is $K \in \mathbb{N}$ such that $\mathbb{Z} \models C(k(\vec{m} - \vec{n}) + \vec{m})$ for all $k \in \mathbb{Z}$ with $|k| \geq K$.*

Proof. Let $\vec{x} = (x_1, \ldots, x_l)$ and consider a negative literal $p(x_1, \ldots, x_l) \neq 0$ of C, where p is a linear polynomial in the variables $x_1, \ldots, x_l$ with coefficients in $\mathbb{Z}$. Let $h_0 \coloneqq \vec{m} - \vec{n}$ and $q(k) \coloneqq p(k \cdot h_0 + \vec{m})$, then q is a linear polynomial in one variable. By the assumptions we have $q(0) = p(\vec{m}) \neq 0$. Hence, there is at most one $k \in \mathbb{Z}$ such that $q(k) = 0$.

Let $p_1(x_1, \ldots, x_l) \neq 0, \ldots, p_r(x_1, \ldots, x_l) \neq 0$ be all the negative literals of C, then we let

$$K = \left| \max \left(\{0\} \cup \bigcup_{i=1}^{r} \{k + 1 \mid m \in \mathbb{Z}, q_i(k) = 0\} \right) \right|.$$

Clearly, the natural number K exists and we have $\mathbb{Z} \models C(k \cdot h_0 + \vec{m}))$ for all $m \in \mathbb{N}$ with $k \geq k_0$. □

Proposition 7.2.4. *Let $C(\vec{x})$ be an $\mathcal{L}_{\mathrm{LA}}$ clause and let $\vec{m}, \vec{n}$ be solutions in $\mathbb{Z}$ of C, then there is $K \in \mathbb{N}$ such that $\mathbb{Z} \models C(k(\vec{m} - \vec{n}) + \vec{m})$ for all $k \in \mathbb{Z}$ with $|k| \geq K$.*

Proof. An immediate consequence of Lemmas 7.2.2 and 7.2.3. □

We summarize the results of this section in the following proposition.

Corollary 7.2.5. *Let $\varphi(x)$ be a p-free $\exists_1$ formula. There exists an $n \in \mathbb{N}$ such that if φ has n solutions in $\mathbb{N}$, then there exists an infinite strictly descending sequence of integers $(k_i)_{i \in \mathbb{N}}$ with $\mathbb{Z} \models \varphi(k_i)$ for all $i \in \mathbb{N}$.*

Proof. Let $\chi_1, \ldots, \chi_k$ be p-free components such that $\vdash \varphi \leftrightarrow \bigvee_{i=1}^{k} \chi_i$. Let $n = k + 1$ and assume that φ has n solutions in $\mathbb{N}$. Then by the pigeonhole principle there is a component $\chi_{i_0} = (\exists \vec{y}) C_{i_0}(x, \vec{y})$ with two solutions in $n_1, n_2 \in \mathbb{N}$ such that $n_1 < n_2$. Thus, $\mathbb{Z} \models C_{i_0}(n_i, \vec{m}_i)$ for $i = 1, 2$ where $\vec{m}_1, \vec{m}_2$ are suitable sequences of natural numbers. Finally, since $n_1 - n_2 < 0$, we are done by applying Proposition 7.2.4 to C_{i_0}. □

7.3 A model of rule based induction

In this section we construct a family of non-standard structures for the language $\mathcal{L}_{\mathrm{LA}}$ and we make use of the results from Sections 7.1 and 7.2 in order to show that these structures are models of the theory

$$(B + \mathrm{B2} + \mathrm{B3}) + \exists_1(\mathcal{L}_{\mathrm{LA}})\text{-IND}^{R-}.$$

Let us start by introducing some terminology about the models of this theory. Since already the theory $[B, \mathrm{Open}(\mathcal{L}_{\mathrm{LA}})\text{-IND}^{R-}]$ proves B1 and $\mathcal{V}$, the models of $(B + \mathrm{B2} + \mathrm{B3}) + \exists_1(\mathcal{L}_{\mathrm{LA}})\text{-IND}^{R-}$ are composed of a copy of the natural numbers—the standard elements—and copies of the integers, which we call the non-standard elements. The elements of the models we construct below are pairs of the form $n^{[m]} = (m, n) \in \mathbb{N} \times \mathbb{Z}$ such that $m = 0$ implies $n \in \mathbb{N}$. If $m = 0$, then the element is a standard element, otherwise it is non-standard and belongs to the m-th copy of the integers. We call m the type of the element and n the value of the element. We start by defining an operation that will allow us to relate the types of the elements.

Definition 7.3.1. *The function $\restriction : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is given by*

$$n \restriction m := \begin{cases} n & \text{if } n \neq 0 \\ m & \text{if } n = 0 \end{cases}.$$

Definition 7.3.2. *Let $I \in \mathbb{N}$, then the $\mathcal{L}_{\mathrm{LA}}$ structure M_I consist of pairs of the form $n^{[m]}$ with $n \in \mathbb{Z}$, $m \in \mathbb{N}$ and $m \leq I$ such that if $m = 0$ then $n \in \mathbb{N}$. Furthermore, we let*

M_I interpret the non-logical symbols as follows

$$0^{M_I} = 0^{[0]},$$
$$s^{M_I}(n^{[m]}) = (n+1)^{[m]}, \text{ for } n^{[m]} \in M,$$
$$p^{M_I}(n^{[m]}) = \begin{cases} (n \dot{-} 1)^{[0]} & \text{if } m = 0, \\ (n-1)^{[m]} & \text{otherwise.} \end{cases}$$
$$n_1^{[m_1]} +^{M_I} n_2^{[m_2]} = (n_1 + n_2)^{[m_1 \uparrow m_2]}.$$

The structure M_0 is isomorphic to the standard model $\mathbb{N}$. Since we are interested in constructing non-standard structures, we will consider mainly the structures M_I with $I \geq 1$.

Lemma 7.3.3. *Let $I \in \mathbb{N}$, then $M_I \models B + \mathrm{B1} + \mathrm{B3} + \mathcal{V}$.*

Proof. Routine. □

The structures M_I with $I \in \mathbb{N}$ and $I > 0$ have the crucial property that $\mathbb{Z}|_{\{s,p,+\}}$ can be embedded into the non-standard parts of M_I. Hence the solutions of 0-free $\exists_1$ formulas in $\mathbb{Z}$ can be transferred into the non-standard chains of M_I.

Lemma 7.3.4. *Let $m, I \in \mathbb{N}$ with $1 \leq m \leq I$, then the function $\iota_m : \mathbb{Z} \to M_I$, $n \mapsto n^{[m]}$ is an embedding of $\mathbb{Z}|_{\{s,p,+\}}$ into $M_I|_{\{s,p,+\}}$. In particular, if $\varphi(x)$ is a 0-free $\exists_1$ formula, then $\mathbb{Z} \models \varphi(n)$ implies $M_I \models \varphi(n^{[m]})$.*

Proof. It is routine to verify that ι_m is an embedding of $\mathbb{Z}|_{\{s,p,+\}}$ into $M_I|_{\{s,p,+\}}$. The second part of the claim follows from the well-known fact that embeddings preserve $\exists_1$ formulas (see for example [Hod97, Theorem 2.4.1]) □

We can now show that the structures M_I satisfy unnested applications of the induction rule $\exists_1(\mathcal{L}_{\mathrm{LA}})\text{-IND}^{R-}$.

Theorem 7.3.5. *Let $I \geq 1$ and T be a sound $\mathcal{L}_{\mathrm{LA}}$ theory. If $M_I \models T$, then $M_I \models [T, \exists_1(\mathcal{L}_{\mathrm{LA}})\text{-IND}^{R-}]$.*

Proof. Let $\varphi(x)$ be an $\exists_1$ formula and assume that φ is T-inductive. Since T is sound, we have $\mathbb{N} \models \varphi(x)$. Now consider an element $n^{[m]} \in M$. If $m = 0$, then $n \in \mathbb{N}$ and by the observation above $\mathbb{N} \models \varphi(\overline{n})$. By the $\exists_1$-completeness of B we have $B \vdash \varphi(\overline{n})$ and therefore $M_I \models \varphi(\overline{n})$. Since $M_I \models \overline{n} = n^{[0]}$ we obtain $M_I \models \varphi(n^{[m]})$.

Now assume that $m > 0$. By Proposition 7.1.16 there exists a 0-free p-free formula φ' and an $N \in \mathbb{N}$ such that $B + \mathrm{B1} + \mathcal{V} \vdash \varphi(s^N(x)) \leftrightarrow \varphi'(x)$. Hence we have $\mathbb{N} \models \varphi'$ and therefore by Proposition 7.2.5 there is an infinite strictly descending sequence of integers $(k_i)_{i \in \mathbb{N}}$ such that $\mathbb{Z} \models \varphi'(k_i)$ for $i \in \mathbb{N}$. By Lemma 7.3.4 we obtain $M_I \models \varphi'(k_i^{[m]})$ for $i \in \mathbb{N}$. Hence there exists $i_0 \in \mathbb{N}$ such that $k_{i_0} + N \leq n$ and $M_I \models \varphi'(k_{i_0}^{[m]})$, thus, $M_I \models \varphi((k_{i_0} + N)^{[m]})$. Since $M_I \models \varphi(x) \to \varphi(s(x))$, we obtain $M_I \models \varphi(k_{i_0} + N + k)^{[m]}$ for all $k \in \mathbb{N}$. In particular, we have $M_I \models \varphi(n^{[m]})$. □

By iterating the argument above we can show that the structures M_I even satisfy nested applications of $\exists_1(\mathcal{L}_{\mathrm{LA}})\text{-IND}^{R-}$.

Corollary 7.3.6. *Let $I \geq 1$ and T be a sound $\mathcal{L}_{\mathrm{LA}}$ theory such that $M_I \models T$, then $M_I \models T + \exists_1(\mathcal{L}_{\mathrm{LA}})\text{-IND}^{R-}$.*

Proof. We proceed by induction on j to show $[T, \exists_1(\mathcal{L}_{\mathrm{LA}})\text{-IND}^{R-}]_j$. For the base case we have $[T, \exists_1(\mathcal{L}_{\mathrm{LA}})\text{-IND}^{R-}]_0 = T$, hence we are done. For the induction step assume that $M_I \models [T, \exists_1(\mathcal{L}_{\mathrm{LA}})\text{-IND}^{R-}]_j$ and observe that $[T, \exists_1(\mathcal{L}_{\mathrm{LA}})\text{-IND}^{R-}]_j$ is sound. Now obtain $M_I \models [T, \exists_1(\mathcal{L}_{\mathrm{LA}})\text{-IND}^{R-}]_{j+1}$ by Theorem 7.3.5. □

Lemma 7.3.7. $M_1 \models \mathrm{B2}$.

Proof. Clearly it suffices to show that $b_1 \restriction b_2 = b_2 \restriction b_1$ for $b_1, b_2 \in \{0, 1\}$. The only interesting case is $b_1 \neq b_2$, that is, $0 \restriction 1 = 1 = 1 \restriction 0$. □

Corollary 7.3.8. $M_1 \models (B + \mathrm{B2} + \mathrm{B3}) + \exists_1(\mathcal{L}_{\mathrm{LA}})\text{-IND}^{R-}$.

Proof. An immediate consequence of Lemma 7.3.7 and Corollary 7.3.6. □

Lemma 7.3.9. *Let $I \geq 1$, then $M_I \not\models E_{k,n,m}$ for $k, n, m \in \mathbb{N}$ with $0 < n < m$.*

Proof. Now observe that $n \cdot k^{[1]} + \overline{(m-n)k} = (nk)^{[1]} + ((m-n)k)^{[0]} = (nk + (m-n)k)^{[1]} = (mk)^{[1]} = m \cdot k^{[1]}$, but $k^{[1]} \neq k^{[0]}$. □

The structures developed in this section allow us to show that the system $(B + \mathrm{B2} + \mathrm{B3}) + \exists_1(\mathcal{L}_{\mathrm{LA}})\text{-IND}^{R-}$ is strictly weaker than $B + \mathrm{B2} + \mathrm{B3} + \exists_1(\mathcal{L}_{\mathrm{LA}})\text{-IND}^{-}$.

Lemma 7.3.10. *Let $I \geq 1$, then $M_I \not\models \exists_1(\mathcal{L}_{\mathrm{LA}})\text{-IND}^{-}$.*

Proof. Let $\chi(x) = \exists y_1, y_2, y_3 \theta(x, y_1, y_2, y_3)$ with

$$\theta(x, y_1, y_2, y_3) \coloneqq x + y_1 \neq x + y_2 \wedge x + (y_3 + y_1) = x + (y_3 + y_2).$$

We will show that $M_I \not\models I_x\chi(x)$. We first show that $M_I \models \chi(n^{[0]})$ for all $n \in \mathbb{N}$. For this it suffices to observe $n^{[0]} + 0^{[0]} = n^{[0]}$, $n^{[0]} + 0^{[1]} = n^{[1]}$ and $n^{[0]} + (0^{[1]} + 0^{[0]}) = n^{[0]} + 0^{[1]} = n^{[1]} = n^{[0]} + (0^{[1]} + 0^{[1]})$. Hence $M_I \models \theta(n^{[0]}, 0^{[0]}, 0^{[1]}, 0^{[1]})$. Now we will show that $M_I \not\models \chi(n^{[m]})$ for $n \in \mathbb{Z}$ and $m > 0$. Let $k_1^{[m_1]}$, $k_2^{[m_2]}$, $l^{[h]} \in M_I$ and assume that

$$n^{[m]} + k_1^{m_1} \neq n^{[m]} + k_2^{[m_2]}, \tag{*}$$

$$n^{[m]} + (l^{[h]} + k_1^{[m_1]}) = n^{[m]} + (l^{[h]} + k_2^{[m_2]}). \tag{†}$$

Since $m > 0$, we have $m \restriction u = m$ for all $u \in \mathbb{Z}$. Hence by (*) we obtain $n + k_1 \neq n + k_2$. By (†) we obtain $n + l + k_1 = n + l + k_2$, thus $k_1 = k_2$. Therefore $n + k_1 = n + k_2$. Contradiction!

By the above we thus have $M_I \models \chi(0)$. Now let $n^{[m]} \in M_I$. If $m = 0$, then we have $M_I \models \chi((n+1)^{[0]})$ hence $M_I \models \chi(n^{[0]}) \to \chi((n+1)^{[0]})$. If $m > 0$, then we have $M_I \not\models \chi(n^{[m]})$ hence $M_I \models \chi(n^{[m]}) \to \chi((n+1)^{[m]})$. Thus $M_I \not\models I_x\chi(x)$. □

An interesting question is whether the theory $B + \mathrm{B2} + \mathrm{B3} + \exists_1(\mathcal{L}_{\mathrm{LA}})\text{-IND}^-$ is conservative for sentences of the form $(\forall x)(\exists \vec{y})\varphi(x, \vec{y})$ with φ quantifier-free over the theory $(B + \mathrm{B2} + \mathrm{B3}) + \exists_1(\mathcal{L}_{\mathrm{LA}})\text{-IND}^{R-}$. A negative answer to that question would provide us with an unprovability result for clause set cycles that is similar to the one given in Section 6.4.4.

Furthermore, the structures M_I with $I \geq 1$ show us that the axiom B2 is not redundant in $(B + \mathrm{B2} + \mathrm{B3}) + \exists_1(\mathcal{L}_{\mathrm{LA}})\text{-IND}^{R-}$.

Lemma 7.3.11. $(B + \mathrm{B3}) + \exists_1(\mathcal{L}_{\mathrm{LA}})\text{-IND}^{R-} \nvdash \mathrm{B2}$.

Proof. By Lemma 7.3.3 we have $M_2 \models B + \mathrm{B3}$. Since $0^{[1]} + 0^{[2]} = 0^{[1|2]} = 0^{[1]} \neq 0^{[2]} = 0^{[2|1]} = 0^{[2]} + 0^{[1]}$, we obtain $M_2 \models (B + \mathrm{B3}) + \exists_1(\mathcal{L}_{\mathrm{LA}})\text{-IND}^{R-} \nvdash \mathrm{B2}$ by Corollary 7.3.6. □

Similarly, we conjecture that the axiom B3 is also not redundant in the theory $(B + \mathrm{B2} + \mathrm{B3}) + \exists_1(\mathcal{L}_{\mathrm{LA}})\text{-IND}^{R-}$.

Conjecture 7.3.12. $(B + \mathrm{B2}) + \exists_1(\mathcal{L}_{\mathrm{LA}})\text{-IND}^{R-} \nvdash \mathrm{B3}$.

7.4 A model of the induction schema

In the previous section we have constructed models that show that $(B + \mathrm{B2} + \mathrm{B3}) + \exists_1(\mathcal{L}_{\mathrm{LA}})\text{-IND}^{R-}$ is a strictly weaker system than $B + \mathrm{B2} + \mathrm{B3} + \exists_1(\mathcal{L}_{\mathrm{LA}})^-\text{-IND}$. Thus, the following independence result of Theorem 6.4.8, which is proved in the remainder of this section, subsumes Theorem 53 of [HV22]:

$$B + \mathrm{B2} + \mathrm{B3} + \exists_1(\mathcal{L}_{\mathrm{LA}})^-\text{-IND} \nvdash E_{k,n,m},$$

where $k, n, m \in \mathbb{N}$ and $0 < n < m$. A proof of this result is in particular interesting for the studies of clause set cycles, since this independence induces an unprovability result, see Section 6.4.2. We proceed as usual by constructing a suitable non-standard model of the language $\mathcal{L}_{\mathrm{LA}}$.

Definition 7.4.1. *Let M be the structure consisting of elements of the form $(m, n) \in \mathbb{N} \times \mathbb{Z}$ with $m = 0$ implies $n \in \mathbb{N}$ and interpreting the language $\mathcal{L}_{\mathrm{LA}}$ as follows*

$$\begin{aligned} 0^M &= (0, 0), \\ s^M((m, n)) &= (m, n + 1), \\ p^M((0, n)) &= (0, n \dot{-} 1), \\ p^M((m, n)) &= (m, n - 1), \\ (m_1, n_1) +^M (m_2, n_2) &= (\max(m_1, m_2), n_1 + n_2). \end{aligned}$$

Lemma 7.4.2. $M \models B + \mathrm{B1} + \mathrm{B2} + \mathrm{B3}$.

Proof. Since $(0,0) \neq (0,1)$, it is obvious that $M \models$ (A1). We have $p^M(0^M) = p^M((0,0)) = (0,0) = 0^M$, hence $M \models$ (A2). Let $(m,n) \in \mathcal{D}(M)$, then $p^M(s^M((m,n))) = p^M((m,n+1))$. If $m = 0$, then we have $n \in \mathbb{N}$, hence $n+1 > 0$ and therefore $(m,(n+1)\dot{-}1) = (m,n)$. If $m > 0$, then we have $(m,n+1-1) = (m,n)$. Hence $M \models$ (A3). Let $(m,n) \in \mathcal{D}(M)$, then we have $(m,n) +^M 0^M = (m,n) +^M (0,0) = (\max(m,0), n+0) = (m,n)$. Hence $M \models$ (A4). Let $(m_1,n_1),(m_2,n_2) \in \mathcal{D}(M)$, then we have $(m_1,n_1) +^M s^M((m_2,n_2)) = (m_1,n_1) +^M (m_2,n_2+1) = (\max(m_1,m_2), n_1+n_2+1) = s^M((\max(m_1,m_2), n_1+n_2)) = s^M((m_1,n_1) +^M (m_2,n_2))$. Hence $M \models$ (A5). Let $(m,n) \in \mathcal{D}(M)$. If $m = 0$, then there are two cases to consider if $n = 0$, then we are done otherwise $n > 0$ and we have $(m,n-1) = p^M(m,n)$ and $s^M(m,n-1) = (m,n)$. If $m > 0$, then we have $s^M(p^M((m,n))) = s^M((m,n-1)) = (m,n)$. Hence $M \models$ B1. Since max and $+$ are associative, clearly $+^M$ is also associative (construction by product). Similarly, $+^M$ is commutative. □

The following structure will be used to work with the left component of the elements of M.

Definition 7.4.3. *Let $\mathbb{M}$ be the $\mathcal{L}_{\mathrm{LA}}$ structure with domain $\mathbb{N}$ and interpreting the non-logical symbols as follows: $0^{\mathbb{M}} = 0$, $s^{\mathbb{M}}(n) = n$, $p^{\mathbb{M}}(n) = n$, and $n_1 +^{\mathbb{M}} n_2 = \max(n_1,n_2)$.*

Definition 7.4.4. *Let $\vec{n} = (n_1,\ldots,n_k)$, $\vec{m} = (m_1,\ldots,m_k)$ be finite sequences of natural numbers, then we denote by $\vec{n} \uparrow \vec{m}$ the finite sequence*

$$((n_1,m_1),\ldots,(n_k,m_k)).$$

Evaluating a p-free term in M can be reduced to evaluating the term over $\mathbb{M}$ and $\mathbb{Z}$.

Lemma 7.4.5. *Let $t(x_1,\ldots,x_l)$ be a p-free term, $\vec{m} = (m_1,\ldots,m_l)$ a sequence of natural numbers, and $\vec{n} = (n_1,\ldots,n_l)$ a sequence of integers such that $(m_i,n_i) \in \mathcal{D}(M)$ for $i = 1,\ldots,l$, then $t^M(\vec{m} \uparrow \vec{n}) = (t^{\mathbb{M}}(\vec{m}), t^{\mathbb{Z}}(\vec{n}))$.*

Proof. We proceed by induction on the structure of the term t. If t is 0, then we have $t^M = (0,0) = (0^{\mathbb{M}}, 0^{\mathbb{Z}}) = (t^{\mathbb{M}}, t^{\mathbb{Z}})$. If t is a variable x_i, then $t^M(\vec{m} \uparrow \vec{n}) = (m_i,n_i) = (t^{\mathbb{M}}(\vec{m}), t^{\mathbb{Z}}(\vec{n}))$. If t is of the form $s(u)$, then u is p-free and therefore we can apply the induction hypothesis to u. We have

$$\begin{aligned} t^M(\vec{m} \uparrow \vec{n}) &= s^M(u^M(\vec{m} \uparrow \vec{n})) = s^M((u^{\mathbb{M}}(\vec{m}), u^{\mathbb{Z}}(\vec{n}))) = (u^{\mathbb{M}}(\vec{m}), u^{\mathbb{Z}}(\vec{n}) + 1) \\ &= (s^{\mathbb{M}}(u^{\mathbb{M}}(\vec{m})), s^{\mathbb{Z}}(u^{\mathbb{Z}}(\vec{n}))) = (t^{\mathbb{M}}(\vec{m}), t^{\mathbb{Z}}(\vec{n})). \end{aligned}$$

If t is $u+v$, then u and v are p-free. Hence we have

$$\begin{aligned} t^M(\vec{m} \uparrow \vec{n}) &= u^M(\vec{m} \uparrow \vec{n}) +^M v^M(\vec{m} \uparrow \vec{n}) = (u^{\mathbb{M}}(\vec{m}), u^{\mathbb{Z}}(\vec{n})) +^M (v^{\mathbb{M}}(\vec{m}), v^{\mathbb{Z}}(\vec{n})) \\ &= (\max(u^{\mathbb{M}}(\vec{m}), v^{\mathbb{M}}(\vec{m})), u^{\mathbb{Z}}(\vec{n}) + v^{\mathbb{Z}}(\vec{n})) = (t^{\mathbb{M}}(\vec{m}), t^{\mathbb{Z}}(\vec{n})). \end{aligned}$$ □

Lemma 7.4.6. *Let $\pi(\vec{x})$ be a p-free atom and $\vec{m} \uparrow \vec{n}$ a finite sequence of elements of M, then $M \models \pi(\vec{m} \uparrow \vec{n})$ if and only if $\mathbb{M} \models \pi(\vec{m})$ and $\mathbb{Z} \models \pi(\vec{n})$.*

Proof. Let π be of the form $u = v$, then we have $M \models \pi(\vec{m} \uparrow \vec{n})$ if and only if $u^M(\vec{m} \uparrow \vec{n}) = v^M(\vec{m} \uparrow \vec{n})$. By Lemma 7.4.5 this is the case if and only if $(u^{\mathbb{M}}(\vec{m}), u^{\mathbb{Z}}(\vec{n})) = (v^{\mathbb{M}}(\vec{m}), v^{\mathbb{Z}}(\vec{n}))$, which is the case if and only if $u^{\mathbb{M}}(\vec{m}) = v^{\mathbb{M}}(\vec{m})$ and $u^{\mathbb{Z}}(\vec{n}) = v^{\mathbb{Z}}(\vec{n})$. Hence $M \models \pi(\vec{m} \uparrow \vec{n})$ if and only if $\mathbb{M} \models \pi(\vec{m})$ and $\mathbb{Z} \models \pi(\vec{n})$. □

Lemma 7.4.7. *Let $t(x_1, \ldots, x_l)$ be a term and $m_1, \ldots, m_l \in \mathbb{N}$, then*

$$t^{\mathbb{M}}(m_1, \ldots, m_l) = \max\left(\{0\} \cup \{m_i \mid i \in \{1, \ldots, l\}, x_i \in \mathit{Var}(t)\}\right)$$

Proof. We proceed by induction on the structure of the term t. Let $\vec{m} = (m_1, \ldots, m_l)$ If $t = 0$, then we have $t^{\mathbb{M}}(\vec{m}) = 0 = \max\{0\}$. If $t = x_i$ with $i \in \{1, \ldots, l\}$, then $t^{\mathbb{M}}(\vec{m}) = m_i = \max\{0, m_i\}$. If $t = s(t')$, then $\mathit{Var}(t) = \mathit{Var}(t')$ and $t^{\mathbb{M}}(\vec{m}) = (t')^{\mathbb{M}}(\vec{m})$. Thus by the induction hypothesis $t^{\mathbb{M}}(\vec{m}) = \max\left(\{0\} \cup \{m_i \mid i \in \{1, \ldots, l\}, x_i \in \mathit{Var}(t)\}\right)$. The case where t is of the form $p(t')$ is analogous. If t is of the form $t_1 + t_2$, then $\mathit{Var}(t) = \mathit{Var}(t_1) \cup \mathit{Var}(t_2)$ and by the induction hypothesis

$$\begin{aligned} t^{\mathbb{M}}(\vec{m}) &= \max\left\{\max\left(\{0\} \cup \{m_i \mid i \in \{1, \ldots, l\}, x_i \in \mathit{Var}(t_j)\}\right) \mid j = 1, 2\right\} \\ &= \max\left(\{0\} \cup \{m_i \mid i \in \{1, \ldots, l\}, x_i \in \mathit{Var}(t)\}\right). \end{aligned}$$ □

We will now also write $n^{[m]}$ to denote the element (m, n). Let $\vec{m} = (m_1, \ldots, m_n)$ be a sequence of integers, then $\vec{m} \dot{+} k$ denotes the sequence $(m_1 + k, \ldots, m_n + k)$.

Lemma 7.4.8. *Let $t(x_1, \ldots, x_l)$ be a term with $\mathit{Var}(t) \neq \varnothing$, then*

$$t^{\mathbb{M}}(\vec{m} \dot{+} k) = t^{\mathbb{M}}(\vec{m}) + k.$$

Proof. An immediate consequence of Lemma 7.4.7 and the properties of max. □

Lemma 7.4.9. *Let $\pi(\vec{x}) \coloneqq u = v$ be an $\uparrow\uparrow$ atom, $\vec{m}$ a vector of natural numbers, and $k \in \mathbb{N}$. Then*

$$\mathbb{M} \models \pi(\vec{m} \dot{+} k) \text{ if and only if } \mathbb{M} \models \pi(\vec{m}),$$

Proof. For the "only if" direction assume that $\mathbb{M} \models \pi(\vec{m} \dot{+} k)$. Then by Lemma 7.4.8 we obtain $u^{\mathbb{M}}(\vec{m}) + k = u^{\mathbb{M}}(\vec{m} \dot{+} k) = v^{\mathbb{M}}(\vec{m} \dot{+} k) = v^{\mathbb{M}}(\vec{m}) + k$. Therefore $u^{\mathbb{M}}(\vec{m}) = v^{\mathbb{M}}(\vec{m})$, that is, $\mathbb{M} \models \pi(\vec{m})$. For the "if" direction assume that $\mathbb{M} \models \pi(\vec{m})$, then we have $u^{\mathbb{M}}(\vec{m}) = v^{\mathbb{M}}(\vec{m})$ and therefore $u^{\mathbb{M}}(\vec{m}) + k = v^{\mathbb{M}}(\vec{m}) + k$. By Lemma 7.4.8 we obtain $u^{\mathbb{M}}(\vec{m} \dot{+} k) = v^{\mathbb{M}}(\vec{m} \dot{+} k)$. Hence $\mathbb{M} \models \pi(\vec{m} \dot{+} k)$. □

Lemma 7.4.10. *Let $\nu(\vec{x})$ be a negative literal, $\vec{m}$ a vector of natural numbers, and $k \in \mathbb{N}$. If $\mathbb{M} \models \nu(\vec{m})$, then $\mathbb{M} \models \nu(\vec{m} \dot{+} k)$.*

Proof. Let ν be $u \neq v$. We consider several cases. If $\mathit{Var}(u) = \varnothing$ and $\mathit{Var}(v) \neq \varnothing$, then we have $u^{\mathbb{M}} = 0$ and by Lemma 7.4.8 we have $v^{\mathbb{M}}(\vec{m} \dot{+} k) = v^{\mathbb{M}}(\vec{m}) + k$. Since $0 = u^{\mathbb{M}} \neq v^{\mathbb{M}}(\vec{m})$, we have $v^{\mathbb{M}}(\vec{m}) > 0$. Hence $u^{\mathbb{M}} < v^{\mathbb{M}}(\vec{m}) + k = v^{\mathbb{M}}(\vec{m} \dot{+} k)$, thus, $\mathbb{M} \models \nu(\vec{m} \dot{+} k)$. The case where $\mathit{Var}(u) \neq \varnothing$ and $\mathit{Var}(v) = \varnothing$ is symmetric. Now assume that $\mathit{Var}(u), \mathit{Var}(v) \neq \varnothing$. Then we are done by Lemma 7.4.9. The case where $\mathit{Var}(u) = \mathit{Var}(v) = \varnothing$ is not possible, since otherwise $u^{\mathbb{M}}(\vec{m}) = 0 = v^{\mathbb{M}}(\vec{m})$. □

Proposition 7.4.11. *Let $\chi(x)$ be a p-free component whose positive literals are $\uparrow\uparrow$. If χ has two solutions $n_1^{[0]}$ and $n_2^{[0]}$ in M with $n_1 < n_2$, then there is $n_0 \in \mathbb{N}$ such that $M \models \chi((m, n(n_2 - n_1) + n_1))$, for all $n \in \mathbb{Z}$ with $|n| \geq n_0$ and $m \geq 1$.*

Proof. Let $\chi(x) = \exists y_1, \ldots, y_l C_\chi(x, \vec{y})$. Let $M \models C_\chi((0, n_i), m_i \uparrow k_i)$ with $m_i \uparrow k_i$ a vector of elements of M and $i \in \{1, 2\}$. We define

$$\begin{aligned} q(n) &:= n(n_2 - n_1) + n_1, \\ q'(n) &:= (m, q(n)), \\ \vec{p}(n) &:= n \cdot (\vec{k_2} - \vec{k_1}) + \vec{k_1}), \\ \vec{p'}(n) &:= (\vec{m_1} \dot{+} m) \uparrow \vec{p}(n). \end{aligned}$$

Observe that since $m > 0$, $q'(n) \in \mathcal{D}(M)$ and $\vec{p'}(n)$ is a vector of elements of M for all $n \in \mathbb{Z}$.

We start by showing that $M \models C_\chi^+(q'(n), \vec{p'}(n))$ for all $n \in \mathbb{Z}$. Let $\pi(x, \vec{y})$ be a positive literal of C_χ. Since $M \models C_\chi((0, n_i), m_i \uparrow k_i)$ for $i = 1, 2$, we obtain by Lemma 7.4.6 $\mathbb{M} \models \pi(0, \vec{m_1})$ and $\mathbb{Z} \models \pi(n_i, \vec{k_i})$ for $i = 1, 2$. Furthermore, by Lemma 7.4.6 and by the definitions of $q'(n)$ and $\vec{p'}(n)$, it suffices to show $\mathbb{M} \models \pi(m, \vec{m_1} \dot{+} m)$ and $\mathbb{Z} \models \pi(q(n), \vec{p}(n))$. Since the positive literals of χ are $\uparrow\uparrow$, π is of the form $u = v$ with $Var(u), Var(v) \neq \varnothing$. Hence by Lemma 7.4.9 we obtain $\mathbb{M} \models \pi(m, \vec{m_1} \dot{+} m)$. By Lemma 7.2.2 we obtain $\mathbb{Z} \models \pi(q(n), \vec{p}(n))$.

Now let us consider a negative literal ν of the form $u \neq v$. Since $M \models \nu((0, n_1), \vec{m_1} \uparrow \vec{k_1})$, we have by Lemma 7.4.6 either $\mathbb{M} \models \nu(0, \vec{m_1})$ or $\mathbb{Z} \models \nu(n_1, \vec{k_1})$.

- If $\mathbb{M} \models \nu(0, \vec{m_1})$, then by Lemma 7.4.10 we have $\mathbb{M} \models \nu(m, \vec{m_1} \dot{+} m)$. Hence, by Lemma 7.4.6 we obtain $M \models \nu(q'(n), \vec{p'}(n))$ for all $n \in \mathbb{Z}$.
- For the other case, let $\lambda(x, \vec{y})$ be the linear function with coefficients in $\mathbb{Z}$ such that $\mathbb{Z} \models u = v \leftrightarrow \lambda(x, \vec{y}) = 0$. Now let $\lambda'(n) := \lambda(q(n), \vec{p}(n))$, then λ' is a linear function. Since $\mathbb{Z} \models \nu(n_1, \vec{k_1})$, we have $\lambda'(0) \neq 0$. Hence there exists at most one $n_\nu \in \mathbb{Z}$ such that $\lambda'(n_\nu) = 0$. Therefore, $\mathbb{Z} \models \nu(q(n), \vec{p}(n))$, that is, $M \models \nu(q'(n), \vec{q'}(n))$ if $n \neq n_\nu$.

Let $\nu_1, \ldots, \nu_r$ be all the negative literals of χ with $\mathbb{M} \not\models \nu(0, \vec{m_1})$. Now we define $n_0 = \max\{|n_{\nu_i}| + 1 \mid i = 1, \ldots, r\}$. Then by the above arguments we clearly have $M \models C_\chi(q'(n), \vec{p'}(n))$, thus, $M \models \chi(q'(n))$ for all $n \in \mathbb{Z}$ with $|n| \geq n_0$. □

Corollary 7.4.12. *Let $\chi(x)$ be a p-free component whose positive literals are $\uparrow\uparrow$ and let $n_1, n_2 \in \mathbb{N}$ such that $M \models \chi((0, n_i))$ for $i = 1, 2$. Then there exists $(k_i)_{i \in \mathbb{N}}$ with $k_i \in \mathbb{Z}$ such that $k_i > k_{i+1}$ and $M \models \chi(k_i^{[m]})$ for all $i \in \mathbb{N}$ and $m \geq 1$.*

Proof. An immediate consequence of Proposition 7.4.11. □

Corollary 7.4.13. *Let $\varphi(x)$ be an $\exists_1$ formula, then there is $N \in \mathbb{N}$ such that $\varphi(s^N(x))$ is equivalent over $B + \mathrm{B1} + \mathcal{V}$ to a 0-free, p-free disjunction of components $\psi(x)$.*

Proof. An immediate consequence of Proposition 7.1.16 and the fact that we can obtain a logically equivalent disjunction of components over the same language. □

Theorem 7.4.14. $M \models B + \mathrm{B2} + \mathrm{B3} + \exists_1(\mathcal{L}_{\mathrm{LA}})^-\text{-IND}$.

Proof. By Lemma 7.4.2 we have $M \models B + \mathrm{B2} + \mathrm{B3}$. Hence, it remains to show that $M \models \exists_1(\mathcal{L}_{\mathrm{LA}})^-$-IND. Let $\varphi(x)$ be an $\exists_1$ formula. By Corollary 7.4.13 there is $N \in \mathbb{N}$ and a 0-free, p-free disjunction of components $\psi(x) \coloneqq \bigvee_{i=1}^{l} \chi_i(x)$ such that $M \models \varphi(s^N(x)) \leftrightarrow \psi(x)$, where the χ_i are components. Now assume that $M \models \varphi(0)$ and

$$M \models \varphi(x) \to \varphi(s(x)). \tag{$\dagger$}$$

Then we have $M \models \varphi(\overline{n})$ for all $n \in \mathbb{N}$. Let $(m, n) \in \mathcal{D}(M)$, we will show that $M \models \varphi((m, n))$. Since $M \models x = 0 \vee (\exists y)x = s(y)$, it suffices to show $M \models \varphi(s^N(x))$. If $m = 0$, then $n \in \mathbb{N}$ and $M \models (0, n) = \overline{n}$. Hence we have $M \models \varphi((0, n))$.

Now let us consider the case where $m > 0$. By the pigeonhole principle, there is $I \in \{1, \ldots, l\}$ and $(n_i)_{i\in\mathbb{N}}$ with $n_i \in \mathbb{N}$ and $n_i < n_{i+1}$ such that $M \models \chi_I(\overline{n_i})$, for $i \in \mathbb{N}$. Since χ_I is 0-free, the positive literals of χ_I are $\uparrow\uparrow$. Hence, we can apply Corollary 7.4.12 in order to obtain an infinite strictly descending sequence of integers $(k_i)_{i\in\mathbb{N}}$ such that $M \models \chi_I((m, k_i))$ for all $i \in \mathbb{N}$. Thus, there is $J \in \mathbb{N}$ such that $k_J \leq n$. We have $M \models \chi_I((m, k_J))$ and therefore $M \models \varphi((m, k_J))$. By a straightforward induction and making use of ($\dagger$) we obtain $M \models \varphi((m, k_J + k))$ for all $k \in \mathbb{N}$. Therefore, we have in particular $M \models \varphi((m, n))$. □

Proof of Theorem 6.4.8. By Theorem 7.4.14 we can work in structure M. We have $n \cdot (1, k) +^M \overline{(m - n)k} = (1, nk) +^M (0, (m-n)k) = (1, nk+(m-n)k) = (1, mk) = m \cdot (1, k)$, but $(1, k) \neq (0, k)$. □

We conjecture that removing B2 and B3 from Theorem 7.4.14 weakens the result.

Conjecture 7.4.15.

- $B + \mathrm{B2} + \exists_1(\mathcal{L}_{\mathrm{LA}})^-\text{-IND} \nvdash \mathrm{B3}$.
- $B + \mathrm{B3} + \exists_1(\mathcal{L}_{\mathrm{LA}})^-\text{-IND} \nvdash \mathrm{B2}$.

One possibility to prove the second point is to take a model where the type component consists of finite sequences of at least two distinct elements. This model would be associative but not commutative and seems to have similar properties as M.

8 Towards theory exploration-based AITP

In this section we will briefly discuss a family of AITP systems that make use of techniques from the subject of theory exploration. Theory exploration is a subject of automated reasoning that focuses on mechanically discovering new concepts, forming conjectures, and proving theorems from a background theory. The theory exploration paradigm is closer to the way human mathematicians work toward a proof by proving intermediary results and introducing new concepts. Theory exploration systems are typically not primarily used to prove theorems but are used to find interesting results. Hence theorem proving technology and theory exploration technology are not rival paradigms but complement each other. For more details we refer the reader to literature on theory exploration [MBS17; Col02; Len76; Buc+06].

Let us now discuss how theory exploration techniques can be put to use to prove inductive theorems. It is well-known and it has been shown at various occasions in this thesis that proving a given formula by induction often does not work by carrying out induction on the formula itself, but often requires induction on a different and sometimes even more complicated formula. Theory exploration techniques may be used to discover intermediary results from which a formula can be proved by pure first-order reasoning. Typically, formulas are found by generating and proving conjectures of using information from failed proof attempts.

Inductive theorem provers usually integrate into their induction mechanism various heuristics that guess lemmas or induction formulas. These differ from theory exploration-based AITP systems in that they proceed in a top-down way and generate new material from the query, whereas theory exploration-based methods proceed bottom up and do not primarily rely on the query φ to produce new lemmas, see [Joh19]. The AITP system described in [Cla+13] may even be used in theory exploration mode in which it runs without a query and outputs formulas that it proved based on the background theory. This may be useful in a proof assistant where the theory exploration system can provide the user with interesting lemmas.

In the following we will briefly discuss the methods described in [Cla+13; VJ15; SI21] and discuss some possible unprovability results based on a coarse abstraction of these methods. There are of course other theory exploration-based systems that have induction capabilities such as IsaCoSy [JDB09] (the predecessor of Hipster, see [MBS17, Section 5.3.1]), MATHsAiD [MBS17], and Theorema [Buc+06] that we do not address explicitly in this section. In [Cla+13] it is observed that this approach works well in practice. The methods described in [Cla+13; VJ15; SI21] proceed by first generating a set of conjectures, equations in [Cla+13; SI21] or Horn clauses [VJ15], and secondly by

attempting to prove these conjectures incrementally by some analytic induction mechanism over the background theory. In particular proved conjectures are added to the background theory and may thus be used to prove other conjectures. These systems generate the conjectures as follows. First, the system fixes a set of terms T over a finite set of variables V. Usually this is the set of terms that do not exceed some fixed size. After that, the system considers a finite number of substitutions $\sigma_1, \ldots, \sigma_n$ that take the variables V to ground terms. Two terms t_1 and t_2 are considered equivalent if for all $i \in \{1, \ldots, n\}$, $t_1\sigma_i$ is equal to $t_2\sigma_i$ over the background theory. Each such equivalence between terms t_1 and t_2 gives rise to a conjecture $(\forall V)t_1 = t_2$. The method described in [VJ15] uses a slightly modified form of this process to produce Horn clause conjectures. The analytic induction prover used by the systems considered in this section are reminiscent of a Hilbert-style induction rule. Since proved conjectures become part of the background theory, the provers thus behaves akin to nested applications of a Hilbert-style induction rule. In particular [Cla+13] relies on the analytic induction prover described in [Ros12]. The systems typically do not extend the language of the input problem so that we can work over the language of the problem or input theory. The setting assumes that terms and more generally ground formulas can be decided over the background theory.

We can abstract the theory exploration systems described above very coarsely based on the structure of the conjectures and by considering the system that extends the background language by all the true formulas of that structure. In the following we will consider abstractions based on universally closed equations, universally closed Horn clauses, and Π_1 formulas. For each type of system we obtain a simple unprovability result for a formula that does not belong to the class of the conjectures considered by the system. These results are relevant for applications where the system is run in theory exploration mode and show some limits on the use of the lemmas discovered by the system. We start with systems that prove equational lemmas. Let $\mathcal{L}$ be language and M a language structure, then we define

$$\mathrm{Eq}(M) = \{(\forall \vec{x})(t_1 = t_2) \mid \ t_1(\vec{x}), t_2(\vec{x}) \text{ are } \mathcal{L} \text{ terms and } M \models t_1 = t_2\}.$$

Lemma 8.0.1. *$B + \mathrm{Eq}(\mathbb{N}_{\mathcal{L}_{\mathrm{LA}}}) \nvdash x \neq s^{k+1}(x)$, for all $k \in \mathbb{N}$.*

Proof. Let M be the structure whose domain is $\mathbb{N} \cup \{\infty\}$ and that interprets the non-logical symbols as follows: $0^M = 0$, $s^M(n) = n$, $s^M(\infty) = \infty$, $p^M(n) = p^{\mathbb{N}}(n)$, $p^M(\infty) = \infty$, $n +^M m = n + m$, and $n +^M \infty = \infty +^M n = \infty +^M \infty = \infty$, where $n, m \in \mathbb{N}$. It is routine to check that $M \models B$. Now let $\theta(\vec{x}) := u = v$ be an equation where $\vec{x} = (x_1, \ldots, x_k)$ are exactly the free variables of θ and $\mathbb{N} \models \theta$. Furthermore, let $\vec{d} = (d_1, \ldots, d_k)$ with $d_1, \ldots, d_k \in \mathcal{D}(M)$. Since $\mathbb{N} \models \theta$, the variable x_i with $i \in \{1, \ldots, k\}$ occurs in u if and only if it occurs in v. If $d_i = \infty$ for some $i \in \{1, \ldots, k\}$ and $t(\vec{x})$ is a term containing x_i, then we have $M \models t(d_1, \ldots, d_k) = \infty$. Hence, we have $M \models u(\vec{d}) = \infty$ and $M \models v(\vec{d}) = \infty$, hence $M \models \theta(\vec{d})$. Otherwise, $\vec{d} \in \mathbb{N}^k$, thus, $M \models \theta(\vec{d})$. □

Hence, proving only equational statements over a comparatively simple background theory is not sufficient to deal with negative literals. Similar empirical observations are made in [Cla+13] and [VJ15]. This situation can be overcome for example by proving

Horn clauses instead of equational atoms. As shown in the following, this type of system also has very simple unprovability results. In particular, such systems, when working over a theory that is also axiomatized by Horn clauses, cannot deal with disjunctive properties.

Theorem 8.0.2. *Let T be a theory axiomatized by Horn clauses and $\theta_1, \ldots, \theta_k$ atoms with $k \geq 1$ such that $T \vdash \bigvee_{i=1}^{k} \theta_i$, then $T \vdash \theta_i$ for some $i \in \{1, \ldots, k\}$.*

Before we provide a proof of the theorem above we prove the following lemma.

Lemma 8.0.3. *Let $\Gamma \Longrightarrow \Delta$ be a sequent with Γ containing possibly universally quantified Horn formulas and Δ a finite set of atoms. If the sequent $\Gamma \Longrightarrow \Delta$ is provable, then there exists $\delta \subseteq \Delta$ with $|\delta| \leq 1$ such that $\Gamma \Longrightarrow \delta$ is provable.*

Proof. Let π be a **G** proof of $\Gamma \Longrightarrow \Delta$ that is free of non-atomic cuts. Moreover, we can assume without loss of generality that L$\rightarrow$ inferences are part of a subproof of the form

$$\dfrac{\dfrac{\overset{\mu_1}{\Pi \Rightarrow \theta_1, \Lambda} \quad \cdots \quad \overset{\mu_n}{\Pi \Rightarrow \theta_n, \Lambda}}{\Pi \Rightarrow \bigwedge_{i=1}^{n} \theta_i, \Lambda}\, \mathrm{R}\wedge \qquad \overset{(\gamma)}{\Pi, \psi \Rightarrow \Lambda}}{\Pi, \bigwedge_{i=1}^{n} \theta_i \rightarrow \psi \Rightarrow \Lambda.}\, \mathrm{L}\rightarrow$$

We proceed by induction on the number of inferences of the proof π and consider the last inference of π. If π consists only of one inference, then π is either Ax, $\bot$, Refl, Eq and we are trivially done. Now let us assume that π consists of more than one inference. If the last inference of π is L$\forall$, LW, RW, LC or RC we are done by the induction hypothesis. If the last inference of π is Cut, then π is of the form

$$\dfrac{\overset{\pi_1}{\Gamma_1 \Longrightarrow \Delta_1, A} \qquad \overset{\pi_2}{A, \Gamma_2 \Longrightarrow \Delta_2}}{\Gamma_1, \Gamma_2 \Longrightarrow \Delta_1, \Delta_2,}\, \mathrm{Cut}$$

where A is an atom, $\Gamma = \Gamma_1 \cup \Gamma_2$, and $\Delta = \Delta_1 \cup \Delta_2$. By applying the induction hypothesis to π_1 we either obtain a proof μ_1 of $\Gamma_1 \Longrightarrow \delta$, where $|\delta| \leq 1$ and $\delta \subseteq \Delta_1$ or $\Gamma_1 \Longrightarrow A$. In the first case we obtain the desired proof by applying left weakening rules so as to obtain the end-sequent $\Gamma \Longrightarrow \delta$. For the second case we apply the induction hypothesis to π_2 in order to obtain a proof μ_2 of $A, \Gamma_2 \Longrightarrow \delta$ for some $\delta \subseteq \Delta_2$ with $|\delta| \leq 1$. Hence, the following is the desired proof:

$$\dfrac{\overset{\mu_1}{\Gamma_1 \Longrightarrow A} \qquad \overset{\mu_2}{A, \Gamma_1 \Longrightarrow \delta}}{\Gamma_1, \Gamma_2 \Longrightarrow \delta.}\, \mathrm{Cut}$$

Finally, we need to consider the case where the last inference of π is an L$\rightarrow$ rule. By the

assumption above the proof is of the form

$$\dfrac{\dfrac{\overset{\pi_1}{\Gamma \Rightarrow \theta_1, \Delta} \quad \cdots \quad \overset{\pi_n}{\Gamma \Rightarrow \theta_n, \Delta}}{\Gamma \Rightarrow \bigwedge_{i=1}^n \theta_i, \Delta}\, \mathrm{R}\wedge \qquad \overset{(\gamma)}{\Gamma, \psi \Rightarrow \Delta}}{\Gamma, \bigwedge_{i=1}^n \theta_i \to \psi \Rightarrow \Delta.}\, \mathrm{L}\to$$

We apply the induction hypothesis to the proofs in $\pi_1, \ldots, \pi_n$ in order to obtain proofs μ_i of $\Gamma \Longrightarrow \delta_i$ such that $\delta_i \subseteq \{\theta_i\} \cup \Delta$ and $|\delta_i| \leq 1$ for $i = 1, \ldots, n$. If there is $i \in \{1, \ldots, n\}$ such that $\theta_i \notin \delta_i$, then μ_i is the desired proof. Otherwise, by applying the induction hypothesis to γ we obtain a proof γ' of $\Gamma, \psi \Longrightarrow \delta$ with $\delta \subseteq \Delta$ and $|\delta| \leq 1$. Then the following is the desired proof

$$\dfrac{\dfrac{\overset{\pi'_1}{\Gamma \Longrightarrow \theta_1} \quad \cdots \quad \overset{\pi'_n}{\Gamma \Longrightarrow \theta_n}}{\Gamma \Longrightarrow \bigwedge_{i=1}^n \theta_i}\, \mathrm{R}\wedge \qquad \overset{\gamma'}{\Gamma, \psi \Longrightarrow \delta}}{\Gamma, \bigwedge_{i=1}^n \theta_i \to \psi \Longrightarrow \delta.}\, \mathrm{L}\to$$

□

Proof of Theorem 8.0.2. Suppose that $T \vdash \bigvee_{i=1}^k \theta_i$, then by the completeness theorem there exists a finite set Γ of universally quantified Horn clauses that are axioms of T such that $\Gamma \Longrightarrow \Delta$ with $\Delta = \{\theta_1, \ldots, \theta_k\}$ is provable. By Lemma 8.0.3 we obtain a proof of $\Gamma \Longrightarrow \theta_i$ for some $i \in \{1, \ldots, k\}$. By the soundness of **G** we thus have $T \vdash \theta_i$. □

This gives us a simple unprovability result for theory exploration-based AITP methods that infer true universally closed Horn clauses. Let $\mathcal{L}$ be a language and M a language structure, then we define

$$\mathrm{HCl}(M) = \{(\forall \vec{x}) C \mid C(\vec{x}) \text{ is an } \mathcal{L} \text{ Horn clause and } M \models C\}.$$

Corollary 8.0.4. *Let $\lfloor \frac{x}{2} \rfloor$ be a unary function symbol whose intended interpretation is integer division by 2. Let the theory $B_{\lfloor \frac{x}{2} \rfloor}$ extend the base theory of additive arithmetic B by the universal closure of the following formulas*

$$\left\lfloor \frac{0}{2} \right\rfloor = 0, \quad \left\lfloor \frac{\overline{1}}{2} \right\rfloor = 0, \quad \left\lfloor \frac{s(s(x))}{2} \right\rfloor = s\left(\left\lfloor \frac{x}{2} \right\rfloor\right).$$

Then $B_{\lfloor \frac{x}{2} \rfloor} + \mathrm{HCl}(\mathbb{N}_{\mathcal{L}(B_{\lfloor \frac{x}{2} \rfloor})}) \nvdash x = 2 \cdot \lfloor \frac{x}{2} \rfloor \vee x = 2 \cdot \lfloor \frac{x}{2} \rfloor + \overline{1}$.

Proof. An immediate consequence of Theorem 8.0.2 and the fact that $\mathbb{N}_{\mathcal{L}(B_{\lfloor \frac{x}{2} \rfloor})} \models B_{\lfloor \frac{x}{2} \rfloor}$. □

This result shows us that when working with a Horn theory, theory exploration-based AITP methods that work over Horn clauses may only prove clauses of which already one literal can be proven. It would also be interesting to relate this result with the induction

schema for literals so as to possibly obtain a connection between theory exploration methods and the systems based on the single-clause induction rules introduced in [RV19; Haj+20]. An interesting related result of Shoenfield that can be mentioned in this context is the following.

Theorem 8.0.5 ([She63, Theorem 2.2]). *Let T be a $\forall_1$ theory, then*

$$T + \mathrm{Open}(\mathcal{L}(T))\text{-IND} \equiv_{\mathrm{Open}(\mathcal{L}(T))} T + \mathrm{Open}(\mathcal{L}(T))\text{-IND}^R.$$

This shows us that at least when restricted to quantifier-free queries a theory exploration-based method based that uses the induction rule over quantifier-free formulas is not weaker than the corresponding system with the induction axiom for quantifier-free formulas. We conjecture that a similar conservativity result holds for the literal induction schema over the Horn clause induction rule.

Conjecture 8.0.6. *Let T be Horn theory, then*

$$T + \mathrm{Literal}(\mathcal{L}(T))\text{-IND} \equiv_{\mathrm{Open}(\mathcal{L}(T))} T + \mathrm{HCl}(\mathcal{L}(T))\text{-IND}^R.$$

Finally, let us briefly consider the more powerful class of theory exploration based systems that detect true $\forall_1(\mathcal{L})$ (or even true $\Pi_1(\mathcal{L})$) sentences where $\mathcal{L}$ is the language of the background theory. When the background theory T is $\Pi_1(\mathcal{L})$ axiomatized, then by Parikh's theorem (see Theorem 6.2.9) the system $T + \Pi_1^{\mathbb{N}}(\mathcal{L})$ is not able to prove the totality of functions that have a $\Delta_0(\mathcal{L})$ definition and that grow asymptotically faster than any $\mathcal{L}$ term. Thus a theory exploration system that proves $\Pi_1(\mathcal{L})$ lemmas would not be able to prove $(\forall x)(\exists y)x \triangleright y$ over the background theory $T_{\triangleright}$ where $\triangleright$ represents the graph of the function that enumerates the triangular numbers (see Definition 4.2.31). In particular, Proposition 6.2.11 readily applies to this setting and yields an unprovability result.

The results mentioned in this section are very coarse and not very specific to a particular method. It would in the future be interesting to inspect the methods described in [Cla+13; VJ15; SI21] more closely to extract the induction rules that are used by these systems. This will allow us to provide results that are more specific to a given method and to consider practically relevant variants of these induction rules. The observations that we gave in this section show that even if a method is able to recognize all the true formulas of a certain structure, for example equations, Horn clauses, or Π_1 formulas, then there are some basic unprovability phenomena. This may of course be unproblematic if these methods are applied only in a sufficiently restricted setting. However, this becomes a problem if a method has more varied applications such as for example in a proof assistant.

The practical methods described in [Cla+13; VJ15; SI21] limit the shape of the conjectures that are proved by the system because testing more complex formulas is time consuming. Generating equational conjectures is comparatively efficient, because evaluating instances of terms naturally induces an equivalence relation that can be used to generate many equational conjectures. In general it would be interesting to explore techniques for testing more complex formulas, possibly even with quantifiers. Moreover, it

would be interesting to investigate how to extract more information from the instances and their proofs. Let us mention here the method described in [EH15] which is able to construct a proof by induction from proofs of instances given that the instance proofs are suitably structured [Ebn21].

Furthermore, it could be interesting to consider the integration of the theory exploration paradigm with saturation-based proving. In particular, we have mentioned in Section 6.6 the possibility to extend saturation systems with a case split rule and to detect inductive dependencies between the branches. This case split rule could also be used to produce ground material that could be fed into a conjecturing mechanism.

9 Conclusion

In this thesis we have developed a technique for the analysis of the limits of AITP systems. We have applied this technique to the saturation-based automated inductive theorem provers described in [RV19; Haj+20] (see Chapter 4), [Cru17] (see Chapter 5), and [KP13; Ker14] (see Chapter 6). Furthermore, we have applied this analysis technique to a simple theory exploration-based AITP systems inspired by the prover described in [Cla+13] (see Chapter 8). The analysis technique developed in this thesis is the simulation of AITP systems in first-order theories with induction. Although simulating an AITP by a first-order theory abstracts many details of the system, the results in this thesis show that such a simulation often permits to isolate crucial logical features that are responsible for the overall expressivity of the system. The most important features isolated by the simulations carried out in this thesis are the presence or absence of induction parameters, rule-like applications of induction and the propositional structure of the induction formulas as well as their quantifier complexity. Furthermore, the simulation of an AITP system in a first-order theory gives access to first-order semantics via Gödel's completeness theorem for first-order logic. In this thesis semantic arguments have proved to be highly valuable in establishing the unprovability of certain elementary properties by various AITP systems. Almost all of the unprovability results developed in this thesis rely on the construction of some first-order structure. The only exceptions are Corollary 8.0.4 which relies on a purely proof-theoretic argument and Proposition 4.3.3 which mostly relies on a proof-theoretic argument.

The unprovability results developed in this thesis show that even relatively recent systems fail to prove some practically meaningful properties even given any amount of time and memory. For example, we have shown that some systems fail to prove: simple forms of cancellation of addition, cancellation of a multiplicative constants, that every natural number is either even or odd, or the totality of simple function definitions. This situation is explained by the observation that firstly AITP systems impose more or less restrictions on the syntactic composition of induction formulas and that secondly such restrictions are severely limiting the logical strength of the possible arguments. Typically restrictions are imposed on the number of induction parameters, the propositional structure of induction formulas, or the quantifier complexity. In saturation-based AITP systems this situation is especially opaque because preprocessing steps and the induction mechanism itself modify the working language. In particular, we have seen two situations in which this behavior results in the paradoxical situation where an AITP system solves a problem but fails to solve an instance of that problem (see Sections 4.3 and 6.4.2). Overall, our results show that AITP systems tend not to adapt well to more general settings, which is in contrast to possible applications such as in proof assistants that require more flexibility.

The observation that the reduction of AITP systems to first-order theories results in elementary unprovability results is interesting, because this technique ignores many details about the underlying calculus. On the one hand, this situation suggests that even much more elementary unprovability results may be obtained by taking into account more details of the AITP system such as the analyticity of the induction mechanism. Indeed Proposition 4.3.3 shows that is the case for a variant of the system considered in [RV19]. This issue is further discussed in Section 9.3. On the other hand, the unprovability results give rise to the question how they can be overcome. The abstraction realized by the simulation shows that the unprovability results obtained in this thesis rely primarily on the weakness of the respective induction mechanisms and not on the integration of that mechanism into the concrete system. Hence, our unprovability results require a substantial improvement of an induction mechanism in order to be overcome. Fortunately, the upper bounds developed in this thesis make it simple to consider various natural extensions of a system by relaxing certain restrictions. These extensions in turn usually give rise to new unprovability results. Thus we obtain increasingly difficult to overcome unprovability results that are challenging for future AITP systems. An interesting question, that we left open, is whether the extensions suggested by our unprovability results can be implemented efficiently. Besides developing stronger unprovability results by considering extensions of systems, it is also useful to transfer unprovability results to stronger systems. Doing so permits to rule out, in advance, certain extension depending on whether they are sufficient to overcome an unprovability. For example, we have shown the unprovability result of Section 6.4.2 for clause set cycles can be overcome by induction parameters, but also that induction parameters cannot be compensated, for example, by allowing nested applications of the induction rule or even the usage of the induction schema. This knowledge will be valuable for designing more powerful inductive theorem provers that avoid certain unprovability results.

The work in this thesis thus contributes to the development of the logical foundations of inductive theorem proving by identifying logical features that impact the power of arguments by induction. Furthermore, the results in this thesis complement existing empirical observations about AITP and, most notably, provide formal unprovability results where previously only the failure of a concrete implementation could be observed. In particular, the unprovability results shown in this thesis back similar empirically observed failures of implementations to prove certain properties and moreover show the necessity to consider certain extensions such as, for example, those described in [Haj+20] and [Haj+21b]. Moreover, our results provide some guidance for the development of more powerful AITP systems by identifying logical features that are responsible for unprovability results. Furthermore, the results in this thesis are definite in the sense that they are independent of platforms, applications, and implementations. In particular, they do not suffer from the reproducibility problems that may be encountered in an empirical setting where AITP systems often become inaccessible or unmaintained over time.

In the following sections we discuss some directions for future work. In Section 9.1 we consider some AITP systems for which an analysis similar to the ones carried out in this thesis would be interesting. After that, we consider in Section 9.2 possible improvements of the techniques used in this thesis. Finally, in Section 9.3 we briefly discuss future work

on the analyticity of AITP systems.

9.1 Analysis of other methods

The results in this thesis have shown that the analysis method developed in this thesis is able to obtain valuable information about AITP systems. As already mentioned in the introduction there is a wealth of methods and approaches for inductive theorem proving. In principle every one can and should be subjected to an analysis such as the one carried out in this thesis in order to develop a high-level overview of the state of the art of the subject of automated inductive theorem proving. In the following we will mention some particular systems and explain why they are interesting targets for analysis.

The methods described in [Dar68], [Biu+86], [Wan17], [EP20] integrate induction into a saturation-based proof system and thus similar to the methods considered in Chapters 4 to 6. Because of this similarity, analyzing these methods can probably make use of some of the techniques used in this thesis. Moreover, saturation-based theorem proving is currently the dominating theorem proving paradigm. Therefore, these methods are especially interesting to analyze. In particular it would be interesting to compare [EP20] with the systems based on clause set cycles considered in Chapter 6, since this system uses some of the concepts seen in the n-clause calculus. Moreover, the method described in [Wan17] seems to be rather similar to Cruanes's calculus considered in Chapter 5.

In Chapter 8 we have briefly considered some preliminary results for systems based on theory exploration. Empirical evidence [Cla+13; SI21] shows that this is a promising paradigm for automated inductive theorem proving. This may be partly due to these system being inherently less analytic and thus also behave more consistently with respect to deductive closure. Thus it may be interesting to analyze these systems in more detail.

Another interesting family of systems to consider are the AITP systems that implement cyclic calculi such as the one described in [BS11]. The theorem prover Cyclist described in [BGP12] implements an analytic variant of the cyclic calculus and [JOR21] implements cyclic equational reasoning. For cyclic provers it may be necessary to develop different techniques because cyclic calculi are in general stronger than structural induction [BT17].

Term rewriting systems play an important role in computer science and in software engineering. Thus it is particularly interesting to analyze AITP methods that integrate induction into term rewriting systems. A prominent method in this area is the term rewriting induction method described [Red90]. This method has been adapted to more varied settings including for example inequalities, see [NN18].

Some software verification tools such as Zeno [SDE12], RAPID [Gle20], Dafny [Lei10; Jia+21] include induction mechanisms in order to prove properties about programs that involve loops or recursion. Analyzing these systems as in this thesis may help to understand their limits and may suggest improvements such as, for example, combinations with other techniques that are well-suited for a certain class of problems specific to the verification of software.

9.2 Extension of results and techniques

In the following we outline some possible directions for future work that focus on the extension and improvement of results and techniques developed in this thesis. Essentially, these are the strengthening of existing results and the development of proof theoretic techniques, the exploration of various forms of induction, and the development of unprovability results based on other inductive datatypes than natural numbers.

In this thesis we have provided a number of unprovability results for AITP systems. First of all, many of these results may be strengthened considerably, see for example Conjecture 6.4.13 and Conjecture 4.2.38. Moreover, our technique for providing unprovability results relies mostly on the use of semantic arguments. In particular, for most of our unprovability results we have manually constructed a structure and shown that it is indeed a model of a certain theory. It could be interesting to investigate up to which point we can make such developments more systematic. Parikh's theorem (see Theorem 6.2.9) is a good example of a result that allows us to obtain a certain type of unprovability result systematically. Furthermore, our semantic approach ignores entirely the structure of proofs produced by AITP systems. This is also related to the analysis of the analyticity of methods that we will discuss in more detail in Section 4.3. Even though the semantic approach we use here is convenient, developing proof theoretic techniques could be valuable in the future. Proof theoretic techniques could allow us to exploit the structure of proofs produced by AITP systems in order to provide unprovability results. Moreover, some general results may be obtained more easily with proof theoretic techniques than with model theoretic techniques. The rather simple result Lemma 8.0.3 together with Corollary 8.0.4 supports this idea especially if combined with possible analogues of conservativity results such as the ones of Shepherdson (see Theorem 8.0.5).

Another direction in which to continue the work in this thesis is to explore in more details the different variants of induction, such as big-step induction, Hilbert-style induction rules, simultaneous induction, restrictions of the propositional structure, quantifier complexity, parameter-free induction and variants that we have not considered in this thesis such as Hilbert-style rules with contexts, occurrences of the induction variable, various instances of parameter-substitution induction [She63], and so on. In this thesis we have considered only a few combinations of these features that were relevant for the methods that we have considered and for some natural extensions thereof. Exploring the separations between systems that combine these features could anticipate unprovability results that become relevant once a suitable upper bound has been provided for a given method. Furthermore, separation results would induce benchmark problems that are specific to a combination of features. Such a benchmark suite would have the advantage that the difficulty of the problems is known to some extent and thus offers more structure to discriminate methods. Moreover, the suite obtained in this way is not dependent on any system or application and would thus complement suites such as [Cla+15] that are more oriented towards properties of functional programs.

Finally, let us recall that all the unprovability results in this thesis are about properties of natural numbers. Such results are practically less interesting than results about more complicated inductive datatypes because properties about natural numbers are often

handled via decision procedures and SMT solving. However, many of the properties we have considered in this thesis have analogues for more complicated types, for which we immediately obtain similar unprovability results. Consider for example the property

$$x + x = x + 0 \to x = 0,$$

which we have considered in the context of clause set cycles (see Section 6.4.2). Now we work over a language of finite lists constructed by a nullary function symbol *nil* representing the empty list and the binary function symbol *cons* that adds an element to the front of a list and let $\frown$ be an binary infix function symbol that denotes the concatenation of two lists. Then one can show by a theory interpretation that the property

$$x \frown x = nil \frown x \to x = nil,$$

is independent of the basic axioms for the list constructors and the append operations together with parameter-free induction for lists over existentially quantified formulas. Considering unprovability on natural numbers gives results that are more elementary in the sense that they do not rely on the more complicated structure of, say, lists. Moreover, methods that behave uniformly across different inductive datatypes, will not perform better on a more complicated type than on simpler types. This observation suggests that already improving natural number induction will result in overall substantial improvements in AITP. However, it may be interesting to consider the more complicated types such as lists and trees. For such types we may find properties that rely on the structure of the type and would thus be even more complicated to prove than their analogous on natural numbers. Let us mention just two examples of properties on lists where this seems to be the case. Recall that big-step induction over natural numbers can be eliminated via conjunction (see Lemma 4.2.27). However, over lists big-step induction for quantifier-free formulas does not seem to be simulated by quantifier-free induction. Moreover, the cancellation property of the concatenation of lists seems to be more complicated, than the cancellation property of addition. The left cancellation property $x \frown y = x \frown z \to y = z$ can easily be proved with quantifier-free induction. However, the right cancellation property $y \frown x = z \frown x \to y = z$, does not seem to be a consequence of induction over quantifier-free formulas.

9.3 Analyticity

For efficiency reasons practical AITP system often do not try all possible induction formulas, but instead attempt induction only on formulas generated during proof search. We informally refer to this behavior as analyticity. The method developed in this thesis only takes into account the analyticity restrictions to a certain extent, namely the restrictions imposed on the shape of the induction formulas such as, for example, their propositional structure or the quantifier-complexity. However, the restriction of using only those formulas that are generated during proof search may cause incompleteness results with respect to the upper bounds that we have derived in this thesis. In this

thesis we have provided only one simple result (see Proposition 4.3.3) that takes into account such an analyticity restriction. This result shows that concrete systems are in some cases quite far away from the upper bounds that we have developed. Hence, an interesting direction for future research is the refinement of the results of this thesis by an analysis of the trade-off between efficiency and analyticity. Such an analysis may also be carried out at different levels of abstraction depending on how many details about the AITP method are assumed. Furthermore, we have mentioned in this thesis that several induction mechanisms and their variants cannot be distinguished by the abstractions that we have provided (see Section 4.2.3). It could be insightful to investigate the possible discrepancies between these rules and to design mechanisms that subsume a certain number of their variants so as to possibly facilitate the implementation and soundness considerations.

In some AITP systems the analyticity restrictions manifest themselves in the shape of the proofs. Such systems are particularly interesting because these may be analyzed with proof theoretic techniques. Examples of such systems are the systems considered in Chapters 4 to 6. Furthermore, this is also the case of the prover Cyclist [BGP12] which is based on the cyclic sequent calculus introduced in [BS11]. This prover proceeds by a bottom-up proof search and essentially produces cut-free proofs. Recently Das [Das20] has shown that in the context of arithmetic that the logical consequences of cyclic proofs containing only Σ_n formulas are contained in the theory $I\Sigma_{n+1}$. In the setting of arithmetic this result already gives us an upper bound for provers such as Cyclist, however this bound may be improved by taking into account the cut-freeness of the proofs output by Cyclist.

Bibliography

[Ada87] Zofia Adamowicz. "Parameter-Free Induction, The Matiyasevič Theorem and $B\Sigma_1$". In: *Logic Colloquium '86*. Ed. by F.R. Drake and J.K. Truss. Vol. 124. Studies in Logic and the Foundations of Mathematics. Elsevier, 1987, pp. 1–8.

[Bee06] Michael Beeson. "Mathematical Induction in Otter-Lambda". In: *Journal of Automated Reasoning* 36.4 (2006), pp. 311–344.

[Bek03] Lev D. Beklemishev. "Quantifier-free induction schema and the least element principle". In: *Proceedings of the Steklov Institute of Mathematics* 242 (2003), pp. 50–67.

[Bek05] Lev D. Beklemishev. "Reflection principles and provability algebras in formal arithmetic". In: *Russian Mathematical Surveys* 60.2 (Apr. 2005), pp. 197–268.

[Bek97a] Lev D. Beklemishev. "Induction rules, reflection principles, and provably recursive functions". In: *Annals of Pure and Applied Logic* 85.3 (1997), pp. 193–242.

[Bek97b] Lev D. Beklemishev. "Parameter free induction and reflection". In: *Computational Logic and Proof Theory. 5th Kurt Gödel Colloquim, KGC'97 Vienna, Austria, August 25-29, 1997 Proceedings* (Vienna, Austria). Ed. by Georg Gottlob, Alexander Leitsch, and Daniele Mundici. Vol. 1289. Lecture Notes in Computer Science. Springer, 1997, pp. 103–113.

[Bek99] Lev D. Beklemishev. "Parameter free induction and provably total computable functions". In: *Theoretical Computer Science* 224.1 (1999), pp. 13–33.

[BGP12] James Brotherston, Nikos Gorogiannis, and Rasmus Lerchedahl Petersen. "A Generic Cyclic Theorem Prover". In: *Programming Languages and Systems. 10th Asian Symposium, APLAS 2012, Kyoto, Japan, December 11-13, 2012. Proceedings*. Ed. by Ranjit Jhala and Atsushi Igarashi. Vol. 7705. Lecture Notes in Computer Science. Springer, 2012, pp. 350–367.

[BHW12] Matthias Baaz, Stefan Hetzl, and Daniel Weller. "On the complexity of proof deskolemization". In: *Journal of Symbolic Logic* 77.2 (2012), pp. 669–686.

[BIS92] Siani Baker, Andrew Ireland, and Alan Smaill. "On the Use of the Constructive Omega-Rule within Automated Deduction". In: *Logic Programming and Automated Reasoning. International Conference LPAR'92, St. Petersburg, Russia, July 15-20, 1992, Proceedings*. Ed. by Andrei Voronkov. Vol. 624. Lecture Notes in Computer Science. Springer, 1992, pp. 214–225.

[Biu+86] Susanne Biundo, Birgit Hummel, Dieter Hutter, and Christoph Walther. "The Karlsruhe Induction Theorem Proving System". In: *8th International Conference on Automated Deduction. Oxford, England, July 27 - August 1, 1986, Proceedings*. Ed. by Jörg H. Siekmann. Vol. 230. Lecture Notes in Computer Science. Springer, 1986, pp. 672–674.

[BL94] Matthias Baaz and Alexander Leitsch. "On Skolemization and Proof Complexity". In: *Fundamenta Informaticae* 20.4 (1994), pp. 353–379.

[BM79] Robert S. Boyer and J Strother Moore. *A Computational Logic*. ACM Monograph Series. Edited by Thomas A. Standish. Academic Press, New York, 1979.

[Bro05] James Brotherston. "Cyclic Proofs for First-Order Logic with Inductive Definitions". In: *Automated Reasoning with Analytic Tableaux and Related Methods. 14th International Conference, TABLEAUX 2005, Koblenz, Germany, September 14-17, 2005. Proceedings*. Ed. by Bernhard Beckert. Vol. 3702. Lecture Notes in Computer Science. Springer, 2005, pp. 78–92.

[BS11] James Brotherston and Alex Simpson. "Sequent Calculi for Induction and Infinite Descent". In: *Journal of Logic and Computation* 21.6 (Dec. 2011), pp. 1177–1216.

[BT17] Stefano Berardi and Makoto Tatsuta. "Classical System of Martin-Löf's Inductive Definitions Is Not Equivalent to Cyclic Proof System". In: *Foundations of Software Science and Computation Structures. 20th International Conference, FOSSACS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings*. Ed. by Javier Esparza and Andrzej S. Murawski. Vol. 10203. Lecture Notes in Computer Science. Springer, 2017, pp. 301–317.

[BT19] Stefano Berardi and Makoto Tatsuta. "Classical System of Martin-Löf's Inductive Definitions is not Equivalent to Cyclic Proofs". In: *Logical Methods in Computer Science* 15.3 (Aug. 2019), 10:1–10:25.

[Buc+06] Bruno Buchberger, Adrian Crăciun, Tudor Jebelean, Laura Kovács, Temur Kutsia, Koji Nakagawa, Florina Piroi, Nikolaj Popov, Judit Robu, Markus Rosenkranz, and Wolfgang Windsteiger. "Theorema: Towards computer-aided mathematical theory exploration". In: *Journal of Applied Logic* 4.4 (2006), pp. 470–504.

[Bun+89] Alan Bundy, Frank van Harmelen, Jane Hesketh, Alan Smaill, and Andrew Stevens. "A Rational Reconstruction and Extension of Recursion Analysis". In: *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence. Detroit, MI, USA, August 1989*. Ed. by N. S. Sridharan. Vol. 1. Morgan Kaufmann, 1989, pp. 359–365.

[Bun+93] Alan Bundy, Andrew Stevens, Frank van Harmelen, Andrew Ireland, and Alan Smaill. "Rippling: A heuristic for guiding inductive proofs". In: *Artificial Intelligence* 62.2 (1993), pp. 185–253.

[CFM11] Andrés Cordón-Franco, Alejandro Fernández-Margarit, and Francisco Félix Lara Martín. "A note on parameter free Π_1-induction and restricted exponentiation". In: *Mathematical Logic Quarterly* 57.5 (2011), pp. 444–455.

[Cla+13] Koen Claessen, Moa Johansson, Dan Rosén, and Nicholas Smallbone. "Automating Inductive Proofs Using Theory Exploration". In: *Automated Deduction - CADE-24. 24th International Conference on Automated Deduction, Lake Placid, NY, USA, June 9-14, 2013. Proceedings*. Ed. by Maria Paola Bonacina. Vol. 7898. Lecture Notes in Computer Science. Springer, 2013, pp. 392–406.

[Cla+15] Koen Claessen, Moa Johansson, Dan Rosén, and Nicholas Smallbone. "TIP: Tons of Inductive Problems". In: *Intelligent Computer Mathematics. International Conference, CICM 2015, Washington, DC, USA, July 13-17, 2015, Proceedings*. Ed. by Manfred Kerber, Jacques Carette, Cezary Kaliszyk, Florian Rabe, and Volker Sorge. Vol. 9150. Lecture Notes in Computer Science. Springer, 2015, pp. 333–337.

[Col02] Simon Colton. *Automated Theory Formation in Pure Mathematics*. Distinguished dissertations. Springer, 2002.

[Com01] Hubert Comon. "Inductionless Induction". In: *Handbook of Automated Reasoning*. Ed. by Alan Robinson and Andrei Voronkov. Vol. 1. 2 vols. Amsterdam: North-Holland, 2001. Chap. 14, pp. 913–962.

[Cru15] Simon Cruanes. "Extending Superposition with Integer Arithmetic, Structural Induction, and Beyond." PhD thesis. École Polytechnique, Palaiseau, France, Sept. 2015.

[Cru17] Simon Cruanes. "Superposition with Structural Induction". In: *Frontiers of Combining Systems. 11th International Symposium, FroCoS 2017, Brasília, Brazil, September 27-29, 2017, Proceedings*. Ed. by Clare Dixon and Marcelo Finger. Vol. 10483. Lecture Notes in Computer Science. Springer, 2017, pp. 172–188.

[Dar68] Jared L. Darlington. "Automatic theorem proving with equality substitutions and mathematical induction". In: *Machine Intelligence 3*. Ed. by Donald Michie. Vol. 3. Machine Intelligence. Edinburgh University Press, 1968.

[Das20] Anupam Das. "On the logical complexity of cyclic arithmetic". In: *Logical Methods in Computer Science* 16.1 (Jan. 2020), 1:1–1:39.

[Dow08] Gilles Dowek. "Skolemization in Simple Type Theory: the Logical and the Theoretical Points of View". In: *Reasoning in Simple Type Theory. Festschrift in Honor of Peter B. Andrews on His 70th Birthday*. Ed. by Christoph Benzmüller, Chad Brown, Jörg Siekmann, and Richard Statman. Studies in Logic, Mathematical Logic and Foundations. College Publications, 2008.

[Ebn21] Gabriel Ebner. "Inductive theorem proving based on tree grammars". eng. PhD thesis. Wien: Technische Universität Wien, 2021.

[EH15] Sebastian Eberhard and Stefan Hetzl. "Inductive theorem proving based on tree grammars". In: *Annals of Pure and Applied Logic* 166.6 (2015), pp. 665–700.

[EP20] Mnacho Echenim and Nicolas Peltier. "Combining Induction and Saturation-Based Theorem Proving". In: *Journal of Automated Reasoning* 64.2 (2020), pp. 253–294.

[Gen54] Gerhard Gentzen. "Zusammenfassung von mehreren vollständigen Induktionen zu einer einzigen". In: *Archiv für mathematische Logik und Grundlagenforschung* 2.1 (1954), pp. 1–3.

[Gle20] Bernhard Gleiss. "Automated Software Verification using Superposition-based Theorem Proving". PhD thesis. Wien, 2020.

[Göd31] Kurt Gödel. "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I". In: *Monatshefte für Mathematik und Physik* 38.1 (Dec. 1931), pp. 173–198.

[Haj+20] Márton Hajdú, Petra Hozzová, Laura Kovács, Johannes Schoisswohl, and Andrei Voronkov. "Induction with Generalization in Superposition Reasoning". In: *Intelligent Computer Mathematics. 13th International Conference, CICM 2020, Bertinoro, Italy, July 26-31, 2020, Proceedings*. Ed. by Christoph Benzmüller and Bruce R. Miller. Vol. 12236. Lecture Notes in Computer Science. Springer, 2020, pp. 123–137.

[Haj+21a] Márton Hajdu, Petra Hozzová, Laura Kovács, Johannes Schoisswohl, and Andrei Voronkov. "Inductive Benchmarks for Automated Reasoning". In: *Intelligent Computer Mathematics. 14th International Conference, CICM 2021, Timisoara, Romania, July 26-31, 2021, Proceedings*. Ed. by Fairouz Kamareddine and Claudio Sacerdoti Coen. Vol. 12833. Lecture Notes in Computer Science. Springer, 2021, pp. 124–129.

[Haj+21b] Márton Hajdu, Petra Hozzová, Laura Kovács, and Andrei Voronkov. "Induction with Recursive Definitions in Superposition". In: *Proceedings of the 21st Conference on Formal Methods in Computer-Aided Design – FMCAD 2021*. Ed. by Ruzica Piskac and Michael W. Whalen. Vol. 2. Conference Series: Formal Methods in Computer-Aided Design. TU Wien Academic Press, 2021, pp. 246–255.

[Haj21] Márton Hajdu. "Automating inductive reasoning with recursive functions". eng. MA thesis. Technische Universität Wien, 2021.

[Her30] Jacques Herbrand. "Recherches sur la théorie de la démonstration". fr. Doctorat d'État. Université de Paris, 1930.

[HKV21] Petra Hozzová, Laura Kovács, and Andrei Voronkov. "Integer Induction in Saturation". In: *Automated Deduction - CADE 28. 28th International Conference on Automated Deduction, Virtual Event, July 12-15, 2021, Proceedings*. Ed. by André Platzer and Geoff Sutcliffe. Vol. 12699. Lecture Notes in Computer Science. Springer, 2021, pp. 361–377.

[Hod97] Wilfrid Hodges. *A Shorter Model Theory*. Cambridge University Press, 1997.

[HP93] Petr Hájek and Pavel Pudlák. *Metamathematics of first-order arithmetic*. Perspectives in mathematical logic. Springer, 1993.

[HV20] Stefan Hetzl and Jannik Vierling. "Clause Set Cycles and Induction". In: *Logical Methods in Computer Science* 16.4 (Nov. 2020), 11:1–11:17.

[HV22] Stefan Hetzl and Jannik Vierling. "Unprovability results for clause set cycles". In: *Theoretical Computer Science* (2022).

[HV23] Stefan Hetzl and Jannik Vierling. "Induction and Skolemization in saturation theorem proving". In: *Annals of Pure and Applied Logic* 174.1 (2023).

[HW18] Stefan Hetzl and Tin Lok Wong. "Some observations on the logical foundations of inductive theorem proving". In: *Logical Methods in Computer Science* 13.4 (Apr. 2018), 10:1–10:26.

[JDB09] Moa Johansson, Lucas Dixon, and Alan Bundy. "IsaCoSy: Synthesis of Inductive Theorems". In: *Workshop on Automated Mathematical Theory Exploration (Automatheo)*. Hagenberg, Austria, 2009.

[Jeř20] Emil Jeřábek. "Induction rules in bounded arithmetic". In: *Archive for Mathematical Logic* 59.3 (May 2020), pp. 461–501.

[Jia+21] Hongjian Jiang, Yongjian Li, Sijun Tan, and Yongxin Zhao. "Encoding Induction Proof in Dafny". In: *International Symposium on Theoretical Aspects of Software Engineering. TASE 2021, Shanghai, China, August 25-27, 2021*. IEEE, 2021, pp. 95–102.

[Joh+14] Moa Johansson, Dan Rosén, Nicholas Smallbone, and Koen Claessen. "Hipster: Integrating Theory Exploration in a Proof Assistant". In: *Intelligent Computer Mathematics. International Conference, CICM 2014, Coimbra, Portugal, July 7-11, 2014. Proceedings*. Ed. by Stephen M. Watt, James H. Davenport, Alan P. Sexton, Petr Sojka, and Josef Urban. Vol. 8543. Lecture Notes in Computer Science. Springer, 2014, pp. 108–122.

[Joh19] Moa Johansson. "Lemma Discovery for Induction - A Survey". In: *Intelligent Computer Mathematics. 12th International Conference, CICM 2019, Prague, Czech Republic, July 8-12, 2019, Proceedings*. Ed. by Cezary Kaliszyk, Edwin C. Brady, Andrea Kohlhase, and Claudio Sacerdoti Coen. Vol. 11617. Lecture Notes in Computer Science. Springer, 2019, pp. 125–139.

[JOR21] Eddie Jones, C.-H. Luke Ong, and Steven J. Ramsay. "CycleQ: An Efficient Basis for Cyclic Equational Reasoning". In: *CoRR* abs/2111.12553 (2021).

[Ker14] Abdelkader Kersani. "Preuves par induction dans le calcul de superposition." PhD thesis. Université de Grenoble, Oct. 2014.

[Kom22] Ján Komara. "Efficient elimination of Skolem functions in LK^h". In: *Archive for Mathematical Logic* 61.3-4 (2022), pp. 503–534.

[KP13] Abdelkader Kersani and Nicolas Peltier. "Combining Superposition and Induction: A Practical Realization". In: *Frontiers of Combining Systems. 9th International Symposium, FroCoS 2013, Nancy, France, September 18-20, 2013. Proceedings*. Ed. by Pascal Fontaine, Christophe Ringeissen, and Renate A. Schmidt. Vol. 8152. Lecture Notes in Computer Science. Springer, 2013, pp. 7–22.

[KPD88] Richard Kaye, Jeff Paris, and Costas Dimitracopoulos. "On parameter free induction schemas". In: *Journal of Symbolic Logic* 53.4 (1988), pp. 1082–1097.

[KRV17] Laura Kovács, Simon Robillard, and Andrei Voronkov. "Coming to Terms with Quantified Reasoning". In: *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages*. POPL 2017. Paris, France: Association for Computing Machinery, 2017, pp. 260–270.

[KV13] Laura Kovács and Andrei Voronkov. "First-order theorem proving and Vampire". In: *International Conference on Computer Aided Verification*. Springer. 2013, pp. 1–35.

[Lei10] K. Rustan M. Leino. "Dafny: An Automatic Program Verifier for Functional Correctness". In: *Logic for Programming, Artificial Intelligence, and Reasoning*. Ed. by Edmund M. Clarke and Andrei Voronkov. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 348–370.

[Len76] Douglas Lenat. "AM: An artificial intelligence approach to discovery in mathematics as heuristic search". PhD thesis. Stanford University, 1976.

[MBS17] Roy L. McCasland, Alan Bundy, and Patrick F. Smith. "MATHsAiD: Automated mathematical theory exploration". In: *Applied Intelligence* 47.3 (2017), pp. 585–606.

[McC10] W. McCune. *Prover9 and Mace4*. `http://www.cs.unm.edu/~mccune/prover9/`. 2005–2010.

[Mil87] Dale Miller. "A Compact Representation of Proofs". In: *Studia Logica* 46.4 (1987), pp. 347–370.

[MZ06] Georg Moser and Richard Zach. "The Epsilon Calculus and Herbrand Complexity". In: *Stud Logica* 82.1 (2006), pp. 133–155.

[Nag19] Yutaka Nagashima. "LiFtEr: Language to Encode Induction Heuristics for Isabelle/HOL". In: *Programming Languages and Systems. 17th Asian Symposium, APLAS 2019, Nusa Dua, Bali, Indonesia, December 1-4, 2019, Proceedings*. Ed. by Anthony Widjaja Lin. Vol. 11893. Lecture Notes in Computer Science. Springer, 2019, pp. 266–287.

[NN18] Takahiro Nagao and Naoki Nishida. "Rewriting induction for constrained inequalities". In: *Science of Computer Programming* 155 (2018). Selected and Extended papers from the International Symposium on Principles and Practice of Declarative Programming 2016, pp. 76–102.

[Par71] Rohit Parikh. "Existence and Feasibility in Arithmetic". In: *The Journal of Symbolic Logic* 36.3 (1971), pp. 494–508.

[Par72] Charles Parsons. "On **n**-Quantifier Induction". In: *The Journal of Symbolic Logic* 37.3 (1972), pp. 466–482.

[Red90] Uday S. Reddy. "Term Rewriting Induction". In: *10th International Conference on Automated Deduction. Kaiserslautern, FRG, July 24-27, 1990, Proceedings*. Ed. by Mark E. Stickel. Vol. 449. Lecture Notes in Computer Science. Springer, 1990, pp. 162–177.

[RK15] Andrew Reynolds and Viktor Kuncak. "Induction for SMT Solvers". In: *Verification, Model Checking, and Abstract Interpretation. 16th International Conference, VMCAI 2015, Mumbai, India, January 12-14, 2015. Proceedings*. Ed. by Deepak D'Souza, Akash Lal, and Kim Guldstrand Larsen. Vol. 8931. Lecture Notes in Computer Science. Springer, 2015, pp. 80–98.

[Rob18] Simon Robillard. "An Inference Rule for the Acyclicity Property of Term Algebras". In: *Vampire 2017. Proceedings of the 4th Vampire Workshop*. Ed. by Laura Kovács and Andrei Voronkov. Vol. 53. EPiC Series in Computing. EasyChair, 2018, pp. 20–32.

[Ros12] Dan Rosén. "Proving equational Haskell properties using automated theorem provers". MA thesis. Master's thesis, University of Gothenburg, 2012.

[Ros84] H. E. Rose. *Subrecursion: Functions and Hierarchies*. Oxford University Press, 1984.

[RV19] Giles Reger and Andrei Voronkov. "Induction in Saturation-Based Proof Search". In: *Automated Deduction - CADE 27. 27th International Conference on Automated Deduction, Natal, Brazil, August 27-30, 2019, Proceedings*. Ed. by Pascal Fontaine. Vol. 11716. Lecture Notes in Computer Science. Springer, 2019, pp. 477–494.

[Sch20] Johannes Schoisswohl. "Automated Induction by Reflection". eng. MA thesis. Technische Universität Wien, 2020.

[Sch88] Ulf R. Schmerl. "Diophantine equations in fragments of arithmetic". In: *Annals of pure and applied logic* 38.2 (1988), pp. 135–170.

[SDE12] William Sonnex, Sophia Drossopoulou, and Susan Eisenbach. "Zeno: An Automated Prover for Properties of Recursive Data Structures". In: *Tools and Algorithms for the Construction and Analysis of Systems*. Ed. by Cormac Flanagan and Barbara König. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 407–421.

[She63] John Cedric Shepherdson. "Non-standard models for fragments of number theory". In: *The Theory of Models. Proceedings of the 1963 international symposium at Berkeley*. Ed. by J.W. Addison, Leon Henkin, and Alfred Tarski. Studies in Logic and the Foundations of Mathematics. North-Holland, 1963, pp. 342–358.

[She64] John Cedric Shepherdson. "A Non-standard Model for a Free Variable Fragment of Number Theory". In: *Bulletin de l'académie polonaise des sciences* XII.2 (1964), pp. 79–86.

[Sho58] Joseph Robert Shoenfield. "Open sentences and the induction axiom". In: *Journal of Symbolic Logic* 23.1 (1958), pp. 7–12.

[Sho67] Joseph Robert Shoenfield. *Mathematical logic*. CRC Press, 1967.

[SI21] Eytan Singher and Shachar Itzhaky. "Theory Exploration Powered by Deductive Synthesis". In: *Computer Aided Verification. 33rd International Conference, CAV 2021, Virtual Event, July 20-23, 2021, Proceedings, Part II*. Ed. by Alexandra Silva and K. Rustan M. Leino. Vol. 12760. Lecture Notes in Computer Science. Springer, 2021, pp. 125–148.

[Sie91] Wilfried Sieg. "Herbrand analyses". In: *Archive for Mathematical Logic* 30.5-6 (1991), pp. 409–441.

[Ste88] Andrew Stevens. "A Rational Reconstruction of Boyer and Moore's Technique for Constructing Induction Formulas". In: *8th European Conference on Artificial Intelligence. ECAI 1988, Munich, Germany, August 1-5, 1988, Proceedings*. Ed. by Yves Kodratoff. ECAI'88. Pitmann Publishing, 1988, pp. 565–570.

[Tar36] Alfred Tarski. "Der Wahrheitsbegriff in den formalisierten Sprachen". In: *Studia Philosophica* 1 (1936), pp. 261–405.

[TS00] Anne Sjerp Troelstra and Helmut Schwichtenberg. *Basic proof theory. Second edition*. 2nd ed. Vol. 43. Cambridge tracts in theoretical computer science. Cambridge University Press, 2000.

[TZ71] Gaisi Takeuti and Wilson M. Zaring. *Axiomatic set theory*. Graduate texts in mathematics. Springer, 1971.

[VJ15] Irene Lobo Valbuena and Moa Johansson. "Conditional Lemma Discovery and Recursion Induction in Hipster". In: *Electronic Communications of the EASST* 72 (2015): *Proceedings of the 15th International Workshop on Automated Verification of Critical Systems (AVoCS 2015)*. Ed. by Gudmund Grov and Andrew Ireland.

[Vor14] Andrei Voronkov. "AVATAR: The Architecture for First-Order Theorem Provers". In: *Computer Aided Verification. 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings*. Ed. by Armin Biere and Roderick Bloem. Vol. 8559. Lecture Notes in Computer Science. Springer, 2014, pp. 696–710.

[Wan17] Daniel Wand. "Superposition: Types and Induction." PhD thesis. Saarland University, Saarbrücken, Germany, 2017.

[Wei01] Christoph Weidenbach. "Combining Superposition, Sorts and Splitting". In: *Handbook of Automated Reasoning*. Ed. by Alan Robinson and Andrei Voronkov. Vol. 2. Amsterdam: North-Holland, 2001. Chap. 27, pp. 1965–2013.